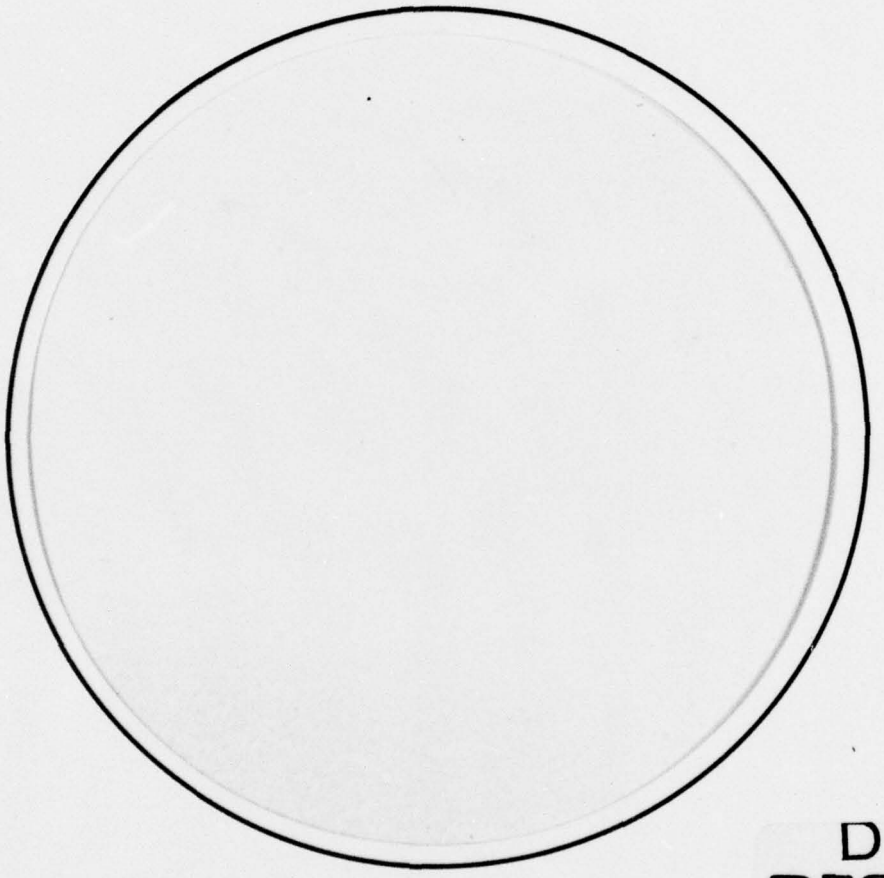
AD-A049 782 ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MASS C--ETC F/G 9/2
COMPLETING THE REQUIREMENTS SET AS A MEANS TOWARDS BETTER DESIG--ETC(U)
DEC 77 R C ANDREU, S E MADNICK N00039-77-C-0255
UNCLASSIFIED CISR-P010-01-06 NL

(12)



Center for Information Systems Research
Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts, 02139
617 253-1000

Contract No. N00039-77-C-0255

Internal Report No. P010-01-06

Deliverable No. A005

(12)

TECHNICAL REPORT #4

COMPLETING THE REQUIREMENTS SET AS A MEANS TOWARDS BETTER DESIGN FRAMEWORKS: A FOLLOW-UP EXERCISE IN SOFTWARE ARCHITECTURAL DESIGN

R. C. Andreu

S. E. Madnick

December, 1977

Principal Investigator:

Prof. S. E. Madnick

Prepared for:

Naval Electronics Systems Command
Washington, D.C. 20360

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(14) CISR-P010-01-06, CISR-TR-4

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Technical Report #4	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
(6) 4. TITLE (and Subtitle): "Completing the requirements set as a means towards better design frameworks: A follow-up exercise in software architectural design"	(9) 5. TYPE OF REPORT & PERIOD COVERED: Technical rept.
6. PERFORMING ORG. REPORT NUMBER: P010-01-06	
(10) 7. AUTHOR(s): Rafael C. Andreu, Stuart E. Madnick	(15) 8. CONTRACT OR GRANT NUMBER(s): N0039-77-C-0255
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Center for Information Systems Research, M.I.T. Sloan School of Management - E53 - 330, Cambridge, MA, 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS	(11) 12. REPORT DATE: Dec [redacted] 77
	13. NUMBER OF PAGES: 80
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

(12) 89p.

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software architectural design; design problem structuring

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The design methodology investigated in this project is applied to the requirements set resulting from the addition of new requirements to the set used previously in ~~Technical Report #3~~. This exercise permits testing the robustness of the methodology and the identification of a more comprehensive DBMS design framework. Guidelines for completing the requirements set are discussed, and the methodology application described, and the results compared with those in Technical Report #3. Finally, a unified methodology framework is presented and a number of areas for further research identified.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409 590

PREFACE

The Center for Information Systems Research (CISR) is a research center of the M.I.T. Sloan School of Management; it consists of a group of Management Information Systems specialists, including faculty memebers, full-time research staff, and student research assistants. The Center's general research thrust is to devise better means for designing, generating and maintaining application software, information systems and decision support systems.

Within the context of the research effort sponsored by the Naval Electronics Systems Command under contract N00039-77-C-0255, CISR proposed to conduct basic research on a systematic approach to the early phases of complex software systems design, one of the main goals being the development of a well defined methodology aimed at explicitly filling the gap between system requirements and program specifications. At the heart of such a methodology is the structuring of the initial set of requirements so as to make apparent the design trade-offs existing among its elements. The main focus of the proposed methodology is the decomposition of that set into subsets of strongly interdependent requirements which would define a meaningful framework for system design. The research project is organized so as to investigate the following four areas:

1) Graph-like representation of requirements sets and suitable decomposition techniques,
2) Design and development of a set of software tools to support the set decomposition activity,
3) Identification of a methodology for the assessment of interdependencies among requirements, as well as guidelines for the interpretation of the obtained decompositions and for the coordination of design subproblems, and
4) Experimental application of the methodology and supporting tools to a specific case, with emphasis on recommendations for their practical use and comparison with more traditional approaches.

This document focuses on the activities carried out at CISR to investigate the fourth area outlined above.

EXECUTIVE SUMMARY

This report describes the application of the design methodology investigated in this project to a new set of requirements. The set employed resulted from the addition of new requirements to that utilized in the analysis discussed in Technical Report #3. The results reported there suggested that certain DBMS design issues were not explicitly addressed by the requirements in that set; this led us to argue that perhaps this was one reason behind the obtainment of rather broad design subproblems in the resulting framework. If this were to be the case, moreover, comparing the results obtained from two similar sets should give us an idea of the robustness of the methodology employed.

To investigate these questions, the follow-up research effort reported herein was undertaken. In particular,

(1) The requirements set was augmented with a number of new requirements found missing in its original version; requirements were added in the areas suggested by the previous analysis and also in others suggested by a DBMS literature search; and

(2) the methodology was applied to the resulting set and a new design framework was identified.

A comparison of the two obtained frameworks points out that while their basic structure is essentially the same --thus giving us evidence regarding the robustness of the methodology--, the one resulting from the augmented set exhibits somewhat better defined subproblems and permits a clearer interpretation of the interactions among them.

The analysis leading to the identification of this second framework is described in detail in the present report. The comparison between the two frameworks is also thoroughly documented and, in addition, an appraisal of our experience with the usage of the methodology is performed. The latter activity permits the identification of more concrete coordination functions which may have to be performed between methodological steps, thus producing a unified methodology framework in which the two design exercises described in Technical Report #3 and in the present one can be succintly represented.

The report concludes that the investigated methodology appears to be capable of producing valuable design frameworks. However, there is a spectrum of areas in which further research has, we believe, great potential to achieve the full incorporation of the methodology to ordinary design practice.

TABLE OF CONTENTS

COMPLETING THE REQUIREMENTS SET AS A MEANS TOWARDS BETTER DESIGN FRAMEWORKS: A FOLLOW-UP EXERCISE IN SOFTWARE ARCHITECTURAL DESIGN.

1. Introduction and Overview

The application of the methodology proposed in [Andreu & Madnick 77] to the early stages of the design of a DBMS was reported in [Andreu 77c]. The result, although meaningful and suggestive of design subproblems which were not apparent in the initial set of requirements, pointed out certain deficiences in that set. This led us to believe that had the analysis been applied to a more complete set, the results might have displayed the structure of the overall design problem more clearly.

Such an outcome was not inconsistent with the objectives of the methodology investigated in the present research project. In fact, being able of detecting pitfalls in the initial set of requirements can be considered a desirable feature for the methodology to exhibit. Nevertheless, it pointed out two related questions which deserve further investigation. On the one hand, there is the primary issue of whether a more complete set of requirements will actually result in a more comprehensive design framework, as we argued it could. On the other hand, if different frameworks in fact result from applying the methodology to two rather similar requirements sets, then the question becomes how different the two frameworks really are; i.e., a question of methodology robustness arises.

To investigate these questions, a follow-up research effort was undertaken. In particular, (1) the requirements set employed in the analysis reported in [Andreu 77c] was augmented with a number of new requirements that were found missing in its original version, and (2) the methodology was applied to the resulting set and a new design framework obtained. This paper reports the results and findings of this follow-up effort.

The report is organized as follows:

- Section 2 describes how the original set of requirements was completed. Emphasis is put in (i) sources of new requirements, (ii) characteristics of the new requirements which are needed in order to keep the resulting augmented set within proper "balance of scope", and (iii) the main areas in which interdependencies between new and old requirements were detected. This section thus concludes with a new graph structure to which the last three steps of the methodology can be applied.

- Section 3 discusses the decomposition analysis performed upon that graph. Using the techniques described in [Andreu 77a], a graph partition is identified; then its main characteristics are discussed vis a vis the structure of the decomposition obtained with the incomplete requirements set ([Andreu 77c]).

- Section 4 interprets the subsets of requirements stemming from the graph decomposition described in section 3 as design subproblems, and investigates their relationships. A new design framework is thus identified which is then compared with that obtained in [Andreu 77c]. Emphasis is put on the main differences between the two frameworks and on the characteristics

of the new requirements set most responsible for them. It is found that while the basic structure is essentially the same for the two frameworks --thus giving us evidence regarding the robustness of the methodology--, the second one, derived from the augmented set of requirements, exhibits somewhat better defined subproblems and permits a clearer interpretation of the interactions among them.

- Section 5 summarizes our experience with the methodology investigated in this project. In retrospect, the results obtained from its two applications are overviewed and an effort made to incorporate their implications in a generalized methodology framework. Finally, we present a brief discussion of a spectrum of areas in which further research has, we believe, great potential to achieve the full incorporation of the methodology to ordinary design practice.

2. Completing the Requirements Set.

As discussed in [Andreu 77c], the results of the analysis reported there suggested that the employed requirements set was incomplete. This conclusion was reached as a consequence of (a) the identification of very broad design subproblems (called "main subproblems", MS's) embracing a wide range of design issues and (b) the need for a second decomposition step on the MS's which led to the isolation of more manageable MSs' components, easily interpreted as design subproblems. This second decomposition step had not been anticipated by the methodology as originally conceived.

These subproblems were congruent with the structure of the overall design problem but they missed some issues that our experience as designers predisposed us to expect. This triggered a search for requirements concerned with such issues in the initial requirements set. The conclusion of set incompleteness was motivated by the failure of such a search.

It was argued ([Andreu 77c]) that the absence of such requirements could be determinant of the lack of concreteness characterizing the obtained MS's. In other words, the possibility of the missing requirements contributing to a clearer identification of design subproblems was suggested. Consequently, we proposed to augment the initial set with such missing requirements and to repeat the analysis.

Realizing that doing simply so, however, might result in still running into incompleteness problems in other areas, it was decided that the set should be completed in a more general way, by adding not only those requirements pointed out as missing by the interpretation of the analysis' re-

sults, but also others that designers' experience could suggest as missing as well. To accomplish this while avoiding personal biases, a DBMS literature search was conducted in an attempt to identify "typical" DBMS requirements not present in the initial set. Thus, there were two main sources of new requirements, as discussed in detail below.

2.1. Sources of New Requirements

The characteristics of the design subproblems identified in [Andreu 77c] pointed out that some requirements specifying the need for different types of data items and others stating that certain queries might be "critical" (i.e., relatively more important than the rest) were in fact missing in the employed requirements set. These, therefore, constituted the first two areas in which new requirements should be added so as to obtain a more complete set. For the reasons outlined above, however, we decided to look for other possibly missing requirements. This was done through a search of the DBMS literature where sets of DBMS requirements have been proposed.

Two main literature sources were found useful to identify other missing DBMS requirements: the publications by Patterson ([Patterson 71] and Huits ([Huits 75]). Although Huits' article is somewhat more specific regarding examples of the proposed requirements, the ones eventually added to the set used in [Andreu 77c] were taken from both.

Patterson and Huits both organize their requirements in categories or classes, then give examples of specific requirements in each class. The classes used by the two are not exactly the same but quite similar; some are present in one, but not in the other. A scanning of the proposed classes pointed out a few which had no "representatives" in the requirements set used in the previous technical report ([Andreu 77c]). These are listed below.

- Possibility of defining and using "keys" on records;
- Specifying that logical structures (in the schema and subschemas) can be combined or operated upon through logical operations;
- Possibility of defining consistency constraints involving records in different files;
- Possibility of defining virtual information;
- Making explicit the need for data independence and a preference for non-redundancy;
- Supporting different types of data items with associated operations permitted among items in a given type;
- Tailoring system to expected query frequency (by type of query);
- Possibility of defining subschemas;
- Facilities to support the DBA activities.

Representative requirements from all these classes were also added to

the requirements set. The specific requirements are discussed in the next section. As a general comment, it is interesting to note that typical DBMS issues such as DBA activities and subschema definition were not mentioned in the original set, as wasn't the call for data independence which, in Patterson's words, should be the "primary [DBMS] design criterion."

2.2. New Requirements

Having identified the main areas in which requirements were missing, we turned to the generation of concrete requirements representing these areas. Two related questions arise at this point:

1) How many new requirements should be generated?

and

2) What level of detail should characterize the new requirements?

Subjective judgement was employed to answer both these questions. This is not to say, however, that requirements were generated arbitrarily. We followed what we thought was a reasonable guideline, which we articulate below. In addition to serve the purpose of helping us decide upon a number of new requirements, this guideline suggests a new desirable property of the employed requirements set that we believe should be added to those proposed in the preceding technical report ([Andreu 77c]). In what follows, we characterize such a guideline by means of some examples.

Focusing on the first area for which new requirements should be generated, that regarding the need for different data items types, two extreme strategies can be taken: (i) Generate a unique requirement in this area; for example, "Different types of data items must be supported," or (ii) Generate a list of specific data types and produce a requirement for each, e.g., "character string data items should be supported," "integer data items should be supported," etc.

In this area, it becomes apparent that the new requirements should be generated by following a scheme close to the former of these strategies, the reason being that other requirements in the original set weren't as specific as the ones the latter strategy is likely to produce. For example, requirement 6 in the original set read:

> "Field definition permits validation of input datum as to acceptable values,"

which doesn't specify what the acceptable values might be or how they may be organized (e.g., as a series of consecutive values given some "standard" sequence, as several such series, as a maximum length constraint, etc.) Following the second of the strategies outlined above would probably result in an "unbalanced" requirements set, from a scope or "level of detail" viewpoint: if the different data types to be supported are specified in detail, then one would expect requirements stating how acceptable values can be defined for each data type to be also present. Following this strategy, thus, could force us to add new requirements in an area which was already covered by the original set. This we believe is unaccep-

table because in a sense it is tantamount to changing the original problem, rather than completing it. Therefore, we attempted to devise requirements of similar scope to others already present in the initial set.

The same line of reasoning applied to the generation of requirements in other areas. For example, the fact that facilities for tuning system performance were already called for by specifications in the original set allowed us to generate a requirement stating that such facilities should be under the control of the DBA.

In summary, we tried to avoid generating requirements either more detailed or of a wider scope than others in related areas of the original set. This suggested such a "balance of scope" as an additional property that the requirements sets employed in the methodology investigated here should exhibit. Requirements not meeting this property can create problems. For example, numerous detailed requirements can be assessed to be interdependent with another of broader scope, thus bringing them all "closer," for set decomposition purposes, than they would otherwise be. This can cause the broader requirement to be assigned to the same subset as the narrower ones, possibly neglecting its interdependencies with others of similar (broad) scope.

With such a guideline in mind, the set of 87 requirements used in [Andreu 77c] (which for ease of reference is reproduced in Appendix A) was augmented by the 16 new requirements listed in Appendix B (to make further references to these new requirements more convenient, they are assigned consecutive numbers starting at 88). This represents an increase of about 18% in the total number of requirements.

2.3. New Interdependencies

Having established the new set of requirements, the next step was to assess interdependencies among new and old requirements. This activity was performed following the same guidelines proposed in [Andreu 77c]; the results are listed in Appendix C. A few comments on the new interdependencies with examples are included below to give the reader a feeling regarding the main areas in which interdependencies were assessed, and to provide an informal preview of how they can change the decomposition results.

As might be expected, requirements 88 and 89 (Appendix B), specifying that keys can be defined on records and that "standard" logical operations can be performed upon the data structures ("files" on the terminology of the initial requirements set) were found heavily interdependent with other requirements establishing schema characteristics; in addition, interdependencies with DDL characteristics were also assessed. As it will be seen in the following sections, however, their interdependencies in the area of schema characteristics turned out to be relatively more important so that these two requirements in fact "pulled" schema design issues into a separate design subproblem, which this time around appeared in the first decomposition performed. Thus, one of the phenomena that we conjectured could take place after completing the requirements set (see [Andreu 77c]) did actually occur in this case. In subsequent sections, this phenomenon and others are discussed in more detail.

Requirement 90, stating the possibility of defining consistency constraints, was found to be involved in a relatively large number of trade-off interdependencies (TI's), mainly regarding consistency checking upon data base update and change.

The explicit statement that virtual information can exist as a consequence of possible algorithmic relationships among data items (requirement 91) turned out to be instrumental in the identification of a subproblem concerned with physical data base design , as will be shown below. Although it exhibited interdependencies in other areas (e.g., as a way to support more data items without running into storage problems or the implication for DDL capabilities to define algorithmically related data items), the former were more determinant in the context of the overall problem.

DBA-related requirements were found interdependent with a variety of other specifications; this, as the decomposition results discussed below will show, determined that they ended up scattered in different subsets, thus highlighting one of the main characteristics of the methodology under investigation: Although the term "DBA" can tend to pull together the set of requirements specifying related issues in the mind of the designer, the explicit assessment process and the formal decomposition procedure combine to break down the apparent coherence of such a series of requirements which might be present in a first designer's perception. The corollary thus is that while subsets of requirements can seem strongly interdependent when viewed in isolation, they need not be so in the context of the entire set,

once design interdependencies with others are identified during the assessment process. This must be seen as one of the methodology's characteristics which make it, we believe, specially sound; being able to break down designers' unjustified biases was, in fact, one of the objectives established at the outset for the methodology.

Appendix D presents the complete set of interdependencies among all the 103 requirements listed in Appendices A and B. Comparing the associated graph with that used in [Andreu 77c] shows that the former has a slightly higher link density (from an average of 5.6 links per nodes we moved up to 6.447, an increase of about 15%). Although at first sight this might seem to imply that the new graph is more globally compact than the old one and therefore that we won't be able to decompose it in more subgraphs than before, this is not the case, as the next sections will show. The reason behind such a behavior is, of course, that although there are relatively more links, they tend to be more concentrated within the eventual subgraphs. The decomposition of this resulting graph is discussed below.

3. Graph Decomposition Results

Using the graph decomposition package described in [Andreu 77b], the graph whose structure is summarized in Appendix D was broken down into subgraphs in the best way we could identify. The results are reproduced in Appendix E. These results are somewhat different from those obtained by a first decomposition of the original requirements set (that is, without completing it as outlined in the preceding section; see [Andreu 77c]). Prior to examining these results in detail, it is interesting to compare them with the old ones from a general standpoint.

The most apparent difference is in the number of subgraphs obtained, which increased from 4 to 11. As a consequence, the sizes of the obtained subgraphs were smaller and exhibited less variation. As shown in Table 1, the average subgraph size (number of nodes) went from 21.75 to 9.37, a decrease of roughly 56%. The standard deviation also decreased, from 15.94 to 4.457, or 74.5%. This suggests that in trying to interpret the subsets of requirements associated with the subgraphs shown in Appendix E, we will probably be less likely to run into the problems that in [Andreu 77c] motivated us to perform a second decomposition step, namely, the identification of rather broad subproblems side by side with others of narrower scope (recall that some of the subgraphs obtained in the first decomposition there weren't decomposed further in the second, thus making such a circum-

stance apparent). As we will discuss below, this is indeed the case, i.e., such a first decomposition can be used directly to derive a design framework in which subgraphs are interpreted as design subproblems.

	With original set	With augmented set
Average subset size	21.75	9.37
Standard deviation	15.94	4.457
Partition measure (Strength; Coupling)	0.8059 (1.0691; 0.2632)	1.779 (3.267; 1.488)

Table 1

Comparison of the Characteristics of Subsets Obtained from the Original Set and Those Resulting from the Augmented Set

There is also another circumstance which suggests that a second decomposition step is not necessary with the results of Appendix E, as follows. Note that the measure associated with that partition is composed of a strength of 3.267 and a coupling of 1.488, yielding a partition evaluation of 1.779. Looking at the relative importance of the coupling component relative to the strength component, we see that the former amounts to about 45.5% of the latter. In comparison, the first decomposition of the original (incomplete) requirements set --[Andreu 77c]-- produced strength and coupling measures of 1.0691 and 0.2632, respectively, yielding a relative coupling weight of about 24.6%. We notice that the relative importance of the coupling component was sig-

nificantly lower in that situation. This can be interpreted in the following way. Obtaining a relatively low coupling value suggests that the subsets in the partition are rather independent of one another, so that one would expect the coordination among the associated design subproblems to be relatively simple. Thus, one can begin to think in terms of further decomposing each subset in isolation, as was proposed and done in [Andreu 77c]. In fact, this was one of the motivations behind performing the second decomposition step in the analysis of the incomplete requirements set. On the other hand, a relatively high coupling value indicates that decomposing each subset further may be inadequate due to the fact that since the interdependencies among subsets are more important, considering each subset in isolation may lose a good deal of information regarding their interactions with the rest of the requirements in the set. Although informal, an analysis along these lines can be used to decide whether a second decomposition step should be executed or not, or at least to have an idea about the information which may be lost if it is performed.

Finally, it is interesting to compare the results presented in Appendix E with those stemming from the original set of requirements after performing the second decomposition step (see [Andreu 77c]). Table 2 shows this comparison. The new decomposition has higher average subset size (40% higher) and also somewhat higher variation of sizes across subsets (7%). This we find intuitively correct in the sense that since the second decomposition step performed on the incomplete requirements set took each subset obtained in the first decomposition (called MS's in [Andreu 77c]) in

isolation from the others, it neglected the interdependencies among MS's. The analysis in Appendix E didn't neglect such interdependencies, which in this case resulted in bringing together requirements that the individual decomposition of MS's had separated.

The implications of the decomposition shown in Appendix E for the purpose of constructing a design framework are discussed in detail in the next section.

	With original set (2 decomposition steps)	With augmented Set
Average subset size	6.69	9.37
Standard deviation	4.16	4.457

Table 2

Comparison of the Characteristics of Subsets Obtained from the Original Set Through 2 Decomposition Steps and those Resulting from the Augmented Set

3.1. General Structure of the Resulting Decomposition

The results of Appendix E are schematically depicted in Figure 1. The different subsets are represented as circles, labelled in correspondence with the subset (cluster) numbers assigned by the decomposition package,

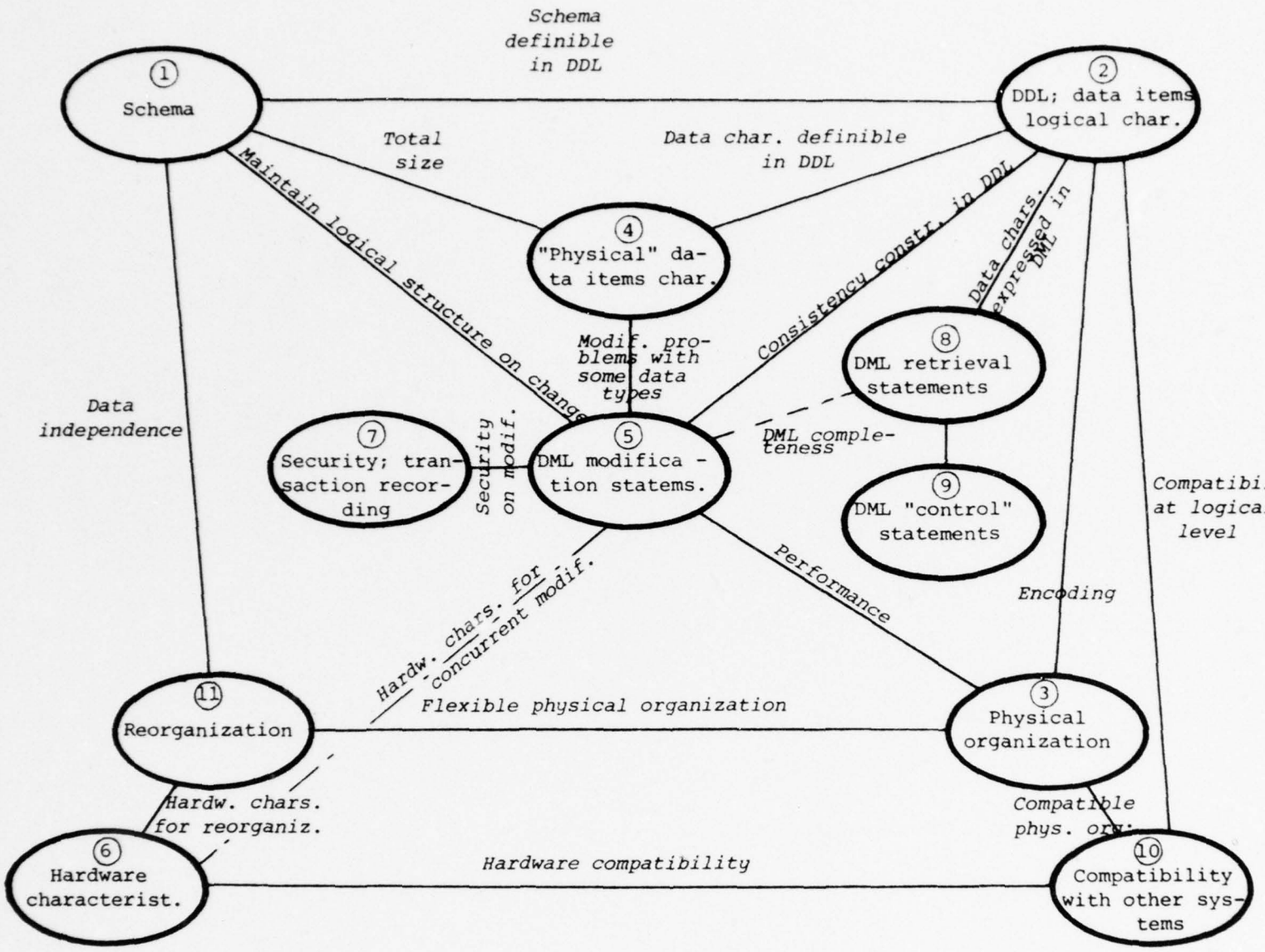

Schema definible in DDL
1 Schema
2 DDL; data items logical char.
Total size
Data char. definible in DDL
Maintain logical structure on changes
4 "Physical" data items char.
Consistency constr. in DDL
Data chars. expressed in DML
Modif. problems with some data types
8 DML retrieval statements
Data independence
7 Security; transaction recording
Security on modif.
5 DML modification statems.
DML completeness
9 DML "control" statements
Compatibility at logical level
Performance
Encoding
Hardw. chars. for concurrent modif.
11 Reorganization
Flexible physical organization
3 Physical organization
Hardw. chars. for reorganiz.
Compatible phys. org.
6 Hardware characterist.
Hardware compatibility
10 Compatibility with other systems

Figure 1.

shown in Appendix E and, in addition, with the associated subproblem names that our interpretation of these subsets as design subproblems suggested. Lines between circles represent the main coordination (coupling) between subproblems (subsets); as discussed below, there were additional coordination lines between the subproblems, but of much less importance than those shown in Figure 1. The specific characteristics of each design subproblem and the relationships among them are discussed in detail in the next section. Here, we examine the general structure of the design framework in an attempt to gain insight into the overall characteristics of the design problem. This will provide the reader with a general overview of the discussion to follow and will allow us to perform a brief comparison of the new framework against that obtained with the incomplete original requirements set, reported in [Andreu 77c].

Figure 1 shows that subproblem 5 plays a sort of "central role" in the resulting framework. The requirements that ended up in subset 5 (see Appendix E) correspond very closely to the same ones which formed MS3 in the first decomposition of the incomplete requirements set; this MS, moreover, was the only one that was not decomposed further in the second decomposition step performed upon that set in [Andreu 77c]. Therefore, the first conclusion drawn from the new analysis is that this subproblem is rather stable and shows up clearly in the two sets of requirements. It corresponds to requirements specifying the characteristics of DML modification statements (add, change, delete), together with requirements made difficult due to such

modifications(e.g.,computations triggered on records related to the ones being modified, integrity maintenance requirements, and the fact that a modification query can be cancelled before completion). One of the new requirements (number 90, calling for the capability of defining consistency constraints explicitly) also ended up in this subset.

This subproblem is central to the design problem given its interactions with others that focus on very different issues. On the one hand, it interacts with subproblems concerned with the logical structure of the data base (subproblems 1 and 2); the nature of these interactions focuses on the need to support modification specifications in the context of the schema. On the other hand, it interacts with subproblems dealing with the physical structure of the data base (subproblem 3 and 6); these interactions emphasize the problems that may arise with the execution of modification requests if the particular physical organization and certain hradware characteristics are not kept in mind. Finally, it interacts with other subproblems which focus on other types of DML statements, thus making explicit the completeness of the eventual DML, and with subproblem 7, concerned with security requirements.

If we compare Figure 1 with the framework resulting from two decomposition steps upon the incomplete set of requirements (Figure 3 in [Andreu 77c]), it is apparent that the central role played by subproblem 5 (there MS3) was already present there.

In the discussion above we have already mentioned the existence of subproblems dealing with logical data base structure (1 and 2, top of Figure 1) and others focusing on physical data base organization (3 and 11,

towards the bottom of Figure 1). Interactions between these two groups of subproblems also exist. In particular (i) the schema definition problem interacts with one (number 3) dealing with physical reorganization as implied by the need for data independence (i.e., reorganization without schema modification), and (ii) the subproblem centered around data items characteristics interacts with the physical organization subproblem as a result of the design having to choose among different alternatives to accomplish data encoding for storage purposes.

Finally, the two subproblems dealing with physical organization interact with the two subproblems at the bottom of Figure 1 (numbers 6 and 10, concerned with hardware characteristics and compatibility requirements).

In summary, and from a general standpoint, if we compare the frameworks that we have identified from (a) the original set of requirements through two decomposition steps and (b) the augmented requirements set through a single decomposition step, there are not too many differences. However, the latter allows a clearer interpretation of the interactions among subproblems since it can be approached more directly (recall that the results reported in [Andreu 77c] required examining the interactions among MS's first, then those among the components of each MS obtained in the second decomposition step, so that the interactions among MS's components in general became rather cumbersome). This will become clearer in the following sections, where each subproblem is analyzed in detail and the interactions among them discussed more thoroughly.

4. The Resulting Design Framework

In this section, the collection of subproblems and interactions depicted in Figure 1 are discussed in more detail and their implications for design are analyzed. First, each subproblem is introduced and commented upon, with emphasis on the requirements that ended up in the associated subsets; then, the graph links corresponding to interdependencies assessments which were cut by the decomposition analysis are taken as basis for a discussion of the design interactions among subproblems.

4.1. Design Subproblems

As shown in Appendix E and depicted in Figure 1, the decomposition of the augmented requirements set resulted in 11 subsets that thus give rise to 11 design subproblems. In this section, each of these subproblems is discussed; for clarity of exposition reasons, we discuss them in roughly a top to bottom order given the organization of Figure 1.

In Figure 1, four main groups of subproblems can be identified:

· Subproblems concerned with data base logical structure and data items characteristics (numbers 1, 2 and 4);

· Subproblems centered around the DML and security issues (numbers 5, 7, 8 and 9);

· Subproblems dealing with data base physical organization issues (numbers 3 and 11); and

· Subproblems focusing on requirements about hardware characteristics and compatibility with other systems (numbers 6 and 10).

Note that there is an approximate correspondence between these groups and the four MS's identified in [Andreu 77c --Figure 3--]. The main differences, as will become clear below, are the separation of physical organization issues from hardware characteristics, and a better defined schema design subproblem whose interactions with others at the logical level are more apparent.

The discussion below is organized in the order implied by these groups of subproblems.

4.1.1. Subproblem 1: Schema

The subset of requirements labelled 1 in Appendix E and Figure 1 corresponds almost exactly to the subproblem labelled C in [Andreu 77c]; there, it emerged as a component of MS1 during the second decomposition step. Here, it showed up in the first decomposition results due to the fact that some of the new requirements contributed to better define it in the context of the overall set. The requirements

included in this subset are the same that defined subproblem C in the previous analysis, with the addition of three new requirements, numbers 88, 89 and 100 (see Appendix B). All these requirements are concerned with the logical structure of the data base and thus contribute to define the schema design problem.

The interpretation of this subset as a design subproblem follows the same lines as the interpretation of subproblem C in [Andreu 77c]. It corresponds to the identification of an appropriate data model and it includes security considerations at the logical level (e.g., requirements 28 through 30, see Appendix A). In addition, it sets forth the need for data base set-up facilities, which interact directly with the initial organization of the data model. The new requirement 100, which establishes the possibility of defining subschemas, ties the problem together from the viewpoint of subschema compatibility with the established schema.

It is interesting to notice that this subproblem emerged partially in the way in which we argued it could in [Andreu 77c]. As will become apparent in the discussion of subproblems' interactions, its relationship with subproblem 2, described in the next section, was clearly separated from others with subproblems that are the counterparts of certain MS 1 components in the previous decomposition of the incomplete requirements set. In addition, however, the inclusion of the three new requirements in this subset contributed to round up this subproblem in a way more clearly apparent than before.

4.1.2. Subproblem 2: Data items' logical characteristics.

This subproblem is the counterpart of subproblem A, a component of MS 2 in the first analysis. It is still centered around the DDL design problem, and establishes the types of data items that are to be supported by the system under design. The identity of this subproblem as compared with that obtained previously has varied in two ways, namely (i) the inclusion of two of the new requirements, numbers 94 and 95 (see Appendix B), further defining the characteristics of the data items by stating that different types may exist and that different associated operations can be established, and (ii) by the fact that requirement 6, which was previously included in this subproblem now is not (it was assigned to subproblem 3 as will be discussed below and it becomes responsible, in the new framework structure, for half the links joining subproblems 2 and 3).

It is interesting to realize that although the two new requirements 94 and 95 were in fact included in the new set as "schema requirements", they had the effect of pulling together subproblem 2, as opposed to contributing to the schema subproblem. This, while perfectly logical, was not anticipated.

As for the new assignment of requirement 6, we will see below that its inclusion in subproblem 3 is meaningful because it can suggest specific

physical organization schemes; its influence upon DDL design is now materialized through the interaction between subproblems 2 and 3.

4.1.3. Subproblem 4: "Physical" data items' characteristics

The requirements in this subset correspond exactly to those in subproblem B of the first analysis ([Andreu 77c]). They focus on data items characteristics mainly concerned with size as seen by the data base user and, although it contributes to further define the DDL design problem discussed above, it has also implications for other subproblems, albeit weaker. For example, its interaction with subproblem 1 is instrumental to the consideration of "logical size" issues for establishing the data base schema. Also, DML modification statements are slightly dependent on the characteristics of the data items involved (e.g., the modification of variable length records may pose additional problems).

4.1.4. Subproblem 5: DML modification statements

This subproblem ties together several others to give the framework of Figure 1. We already discussed some of its characteristics in section 3.1 above. As advanced there, it constitutes the exact counterpart of

MS3 in the results of the analysis performed with the incomplete set of requirements. As it was the case in that analysis, the obtention of this subproblem, including only part of the DML statements that must be supported, is mainly due to its interactions with other subproblems, as opposed to the requirements in the corresponding subset implying a clearly separated design subproblem.

The issues brought together by this subproblem are the need for DML modification statements and its implications regarding the fact that changing certain data items can trigger computations to change other, related ones; this can complicate processing specially if modification requests can be cancelled before completion (note that the requirement specifying this circumstance --number 51, see Appendix A-- is also included in this subproblem). Finally, it must be noticed that the new requirement 90 (stating the existence of consistency constraints, see Appendix B) also takes part in the definition of this subproblem. Its inclusion is relevant given that such constraints must be enforced not only upon the data items explicitly modified by some query, but also upon other modifications done indirectly as a result of triggered computations on related data items.

Thus, the existence of this subproblem responds to the existence of a collection of requirements interacting in such a way that (i) they generate implications for other subproblems and (ii) although touching upon a series of apparently unrelated design issues, point out a number of design trade-offs which the designer should keep in mind. Its identification examplifies the same methodology characteristic noted

before , namely, that interdependencies defined in the context of the overall requirements set may in fact override, for design purposes, others more commonly perceived by the designer which, if taken in isolation, can bias the design process in undesirable ways.

4.1.5. Subproblem 8: DML retrieval statements

This subproblem is the exact counterpart of subproblem K in the analysis reported in [Andreu 77c]. Exactly the same requirements ended up in subproblem 8. This brings up the question of why such a subset of requirements was identified at the first decomposition step in the second analysis while a second step (to decompose MS 4) was necessary in the first, especially if, as is the case, no new requirements are included in the subset. The answer to this question is found when analyzing subproblem 9 (see the next section), the counterpart of the other MS4 component identified in the first analysis. As it turns out, two of the new requirements were assigned to that other subproblem, thus contributing to its clear identification so that, as a byproduct, the remaining of the MS is left as a separate subproblem. In addition to specifying the characteristics of DML selection statements (or the characteristics of data items' references as issued by the user), this subproblem is also concerned with other data base interactions which, although not genuinely DBMS processing, can be similarly implemented (the result of concurrency interdependent requirements), for example, accounting information reporting.

4.1.6. Subproblem 9: DML "control" statements

As anticipated above, this is the counterpart of subproblem L in the previous analysis (i.e., the other component of MS4). Two new requirements ended up in the corresponding subset, namely requirements number 102 and 103, specifying control capabilities of the DBA. Curiously, although these new requirements could at first sight seem only remotely related to the control facilities (which are to be used by the DBA), the analysis of their interactions with the rest of the requirements in the subset points out a deeper design issue that contributes to delineate the identity of this design subproblem: the fact that some of the control facilities should be restricted to DBA usage. For example, a user shouldn't be allowed to suppress the listing of another user, while the DBA, if necessary, can take such an action.

4.1.7. Subproblem 7: Security and transaction recording

The appearance of two new requirements concerned with other DBA activities (numbers 98 and 99, see Appendix 3) contributed to the separation of yet another subproblem which was also identified during the second decomposition step in the first analysis (subproblem F, a component of MS1), concerned with security and the need for transaction recording. In our previous discussion of this subproblem ([Andreu 77c]) we argued that the

presence of sign-on security and transaction recording requirements in the same subproblem made sense because both operations need an analysis of the transaction (request) to be processed and,furthermore,that it suggested to perform transaction recording and security checking at the logical level. This is reinforced in the new results by the inclusion of DBA authorization control and DBA access to accounting information. Authorization control and security checking are certainly related activities that can be performed at the same point in time; moreover, since authorization can be denied on the basis of accounting information, this contributes to tie up the requirements in this subproblem in a stronger fashion.

Thus, this subproblem constitutes another case of new requirements pulling together a subset which was not made apparent as directly in the previous analysis. Furthermore, this particular outcome was not expected from the new set of requirements as strongly as others were (note that the two new requirements contributing to the identification of this subproblem were not suggested by the analysis in [Andreu 77c], but rather by the literature search described at the beginning of this report).

4.1.8. Subproblem 3: Physical organization

Although the name assigned to this subproblem coincides with that assigned to subproblem H in the results of the analysis performed on the incomplete set of requirements, they differ to a greater extent than other

pairs of "corresponding" subproblems discussed above. The main reason for this was,as we illustrate below, the existence of new requirements that hinge more directly in the physical organization problem.

Requirements 64, 73 and 74 (see Appendix A) remained in the subset associated with this subproblem. Constraints on response time and on maintenance time are key ingredients to the physical organization problem. In addition, four new requirements were assigned to this subproblem, numbers 91, 93, 96 and 97 (see Appendix B). Of these, one was expected to behave in this way, namely requirement 97, establishing the existence of critical queries for which response time should be specially controlled (in fact, recall that in [Andreu 77c] such a requirement was found missing during the interpretation of design subproblems and that we argued that its inclusion in the requirements set would perhaps contribute to a clearer identification of the physical organization subproblem, as has been the case). The other three new requirements included in this subproblem were added to the initial set not because they were suggested by the results of the first analysis but rather because the literature search mentioned at the beginning of this report indicated that they were missing as well. Of these three, one (number 93) deals with the physical organization very directly, as it calls for minimizing data redundancy. Requirement number 96 is concerned with access methods more than physical organization per se but, since no separate "access methods subproblem" was identified, its assignement to this subproblem makes good sense. We believe that the no identification of a separate access methods subproblem indicates that it is so closely related to physical organization issues

given the initial requirements set that they are in fact merged together (after all, response time constraints play an important role in both areas and they are at the heart of the present subproblem). The fourth of the new requirements assigned to this subproblem states the possibility of maintaining virtual information; it ties nicely with that calling for the minimization of data redundancy and thus opens physical organization alternatives which weren't made apparent before.

In addition to these four new requirements, this subproblem also included four more which were in fact present in the original incomplete set. Trying to understand why these four were re-assigned in the new results provides further insight into the structure of the framework depicted in Figure 1. Take for example requirement 6 (see Appendix A), which establishes the existence of "domains" -- ranges of acceptable values -- associated with the different data items. In the framework identified in the first analysis, this requirement was assigned to the subproblem dealing with the DDL (the counterpart of subproblem 2 in the new framework). It being switched over to the physical organization subproblem is a consequence, we believe, of two circumstances that are rather difficult to observe in the context of the overall requirements set. On the one hand, the new requirements 94 and 95 contributing to the identification of the DDL subproblem (see 4.1.2 above) permits requirement 6 to be assigned more freely to other subproblems with which it is inter-dependent; on the other hand, this requirement can suggest efficient storage encoding techniques for the different data items (e.g., a very

small range of acceptable values can permit storing data items in such a way that storage is used only to strictly cover the range, even if the values are scattered throughout a wider more "natural" range). Thus, it seems like the decomposition analysis (a) "needs" to identify a DDL subproblem and (b) requirement 6 is used to define such a subproblem only if there are no alternative ways of doing so. When the new requirements 94 and 95 provide one such alternative, requirement 6 is re-assigned to another design subproblem to which it is also relevant.

A similar example is provided by requirement 79, which calls for capability of re-allocating released storage areas. This is an issue that should be taken into account while designing the physical organization, although it can be useful to support its reorganization. In the previous analysis, this requirement was included in the so-called "reorganization" subproblem; in the new analysis, the fact that such a subproblem can be identified by other means permits this requirement to be assigned here.

A related question made apparent by the analysis of this subproblem is what happened to other requirements that were in the subset associated with this subproblem in the results of the previous analysis and are now absent from that subset. An answer to this question can be devised as a consequence of the discussion above. Since this subproblem can be defined by other means, requirements employed in its definition before can be now more freely assigned. Of them, some went to the hardware-centered subproblem; others were switched over to the reorganization subproblem . The latter circumstance, when considered together with changes taking place in the opposite direction, suggests that the coupling

between the physical organization and reorganization subproblems is probably rather high; this is actually true as will be discussed in section 4.2.

4.1.9. Subproblem 11: Reorganization

Several issues concerning this subproblem have already been touched upon in the preceding section, these two subproblems being very closely interrelated.

The subproblem is centered around a set of old requirements that deal mainly with tuning facilities; tuning can be accomplished by physical reorganization. However, it is one of the new requirements that gives identity to this subproblem and which, furthermore, contributes to make its coupling with subproblem 1 (the schema subproblem) more apparent. It is requirement number 92 (see Appendix B), calling explicitly for data independence. This requirement makes it clear that reorganization for tuning purposes shouldn't disrupt the data base logical organization and, consequently, adds to the coupling between this subproblem and the schema design subproblem.

4.1.10. Subproblem 10: Compatibility with other systems

This subproblem corresponds almost exactly to that labelled subpro-

blem H (a component of MS 1) in the previous analysis.

One slight difference is the inclusion of requirement 80 along with requirements 70, 71 and 72 (calling explicitly for system compatibility with others in use by the same Government Agency). Requirement 80 states that file sizes should be limited only by storage capacity. Its inclusion in this subproblem can be interpreted by realizing that compatibility requirements may pose constraints upon the storage facilities needed (e.g., files from other systems might have to be stored on the new system). The major difference , however, concerns the relationships of this subproblem with others, rather than the subproblem itself. In the old analysis, this subproblem showed only a relationship with the physical organization problem; the two, recall, were identified at the same time as two components of MS 1 during the second decomposition step. In the new framework (see Figure 1) this subproblem exhibits a strong relationship with subproblem 2, concerned with data items' logical characteristics. This indicates that compatibility issues should be taken into account at both the physical and logical levels. Such a relationship was precluded from emerging explicitly in the first analysis by the fact that the two subproblems involved were components of two different MS's. Since MS's were decomposed in isolation, relationships among their components were difficult to track down. Thus, completing the requirements set resulted not only in better identification of design subproblems, but also in a more comprehensive display of the relationships among subproblems.

4.1.11. Subproblem 6: Hardware characteristics

This subproblem corresponds very closely to the merger of three subproblems identified in the previous analysis, namely subproblems E, G and J (see [Andreu 77c]). These three subproblems were related to one another by links that we labelled "appropriate hardware." Thus, it is not surprising to see them collapsed together in the new framework. The reason behind such an outcome is probably the fact that since the old three subproblems were obtained through the decomposition of an MS in isolation (MS1), they missed interdependencies with other subproblems which, although weak, can be enough to determine further sources of similarity between them, thus contributing to their merger.

In addition to the requirements assigned before to those three subproblems, three others were included. The first, requirement 67 (see Appendix A), is concerned with having removable disks available, a clear hardware requirement. In the old framework, this requirement was assigned to the physical organization subproblem. This subproblem being now more clearly identifiable through new requirements, requirement 67 is used to further define the hardware problem.

Finally, the two requirements regarding costs (development and maintenance) were assigned to this subproblem. As it happened with the previous analysis, their assignment is the most difficult to interprete. In a sense, costs constraints are relevant to every subproblem . Maybe

this indicates that these two requirements are too global and should be broken down into costs constraints for different system facilities.

4.1.12. Summary

In summary, most of the issues that generated subproblems in the previous analysis did so in the new results as well. The main differences that we detected were (a) a better characterization of design subproblems made possible by the added requirements and indirectly by allowing reassignment of requirements to subproblems where they are relevant and (b) a clearer display of the relationships among subproblems, discussed in more detail below. It seems that both are improvements over the results obtained previously.

4.2. Relationships Among Subproblems

The framework depicted in Figure 1 shows the main interactions among the 11 subproblems described above. In this section we discuss the basic nature of these interactions, as implied by the graph links that remained outside the subsets of requirements in the decomposition shown in Appendix E. A listing of these links is provided in Appendix F, where requirement numbers correspond to those assigned in Appendices A and B.

The most significant interactions among subproblems were identified by computing the coupling existing between each pair of requirements subsets. As Appendix F shows, several such couplings were zero (i.e., no links exist joining certain pairs of subsets). Other couplings were very close to zero (e.g., 0.008 between subproblems 1 and 6); these are not depicted in Figure 1 and are not discussed below due to their relatively low relevance. The following discussion is organized around the structure depicted in Figure 1, in an approximate top to bottom order.

The relationship between subproblems 1 and 2 (schema and DDL) emphasizes the fact that the DDL should allow the definition of the logical constructs present in the schema and, further, the definition of both subschemas and the establishment of security privileges. It is interesting to note that such a relationship, although necessary, wasn't as clearly identified between the counterparts of those two subproblems resulting from the first analysis, partly because they emerged from the

decomposition of different MS's and partly because two of the new requirements actually take part in its definition (requirements 88 and 100, see Appendices B and F).

The relationship between subproblems 1 and 4 (schema and "physical" data items' characteristics) is basically concerned about the total size implied by data items characteristics when put together in the context of the schema, thus bringing relevant size constraints into the schema definition subproblem.

Subproblems 2 and 4 (DDL and data items' "physical" characteristics) interact as one would expect: the data characteristics set forth by the latter should be definable through the DDL, thus indicating that DDL facilities should be available to specify characteristics such a variable sized records (attributes) or multi-valued attributes (repeated fields).

The interaction between subproblems 4 and 5 ("physical" data items' characteristics and DML modification statements), is another relationship that wasn't as clearly identified in the previous analysis, due again to the circumstance that these two subproblems were part of different MS's. It makes explicit the fact that modifying certain data items may pose additional problems which should be taken into account (for example, changing a variable length data item may require more complicated processing than changing fixed length fields).

Subproblem 5 interacts with 2 (DDL) by way of pointing out that integrity and consistency constraints, specially relevant to the processing of modification queries, should be expressible in the DDL.

Between subproblems 2 and 8 (data items logical characteristics and DML retrieval statements), the interaction indicates that the DML should allow references to all existing types of data items, in a well defined way. This brings additional constraints to the DML design problem in the sense that, for example, literal values of different types should be appropriately differentiated (e.g., quotes around character strings to give an instance of an implementation scheme often used).

The relationships between subproblems 8 and 9 (DML retrieval and "control" statements) is one of DML completeness; in particular, it makes explicit the need for control statements in the DML, an issue often neglected in discussions about DML's but which is nevertheless very necessary in almost all non-trivial DBMS processing. In addition, it also points out certain similarities that may exist with the processing of traditional queries, for example, options (e.g., save output for later listing) and certain control requests (e.g., suppression and later generation of listing).

The interaction between subproblems 5 and 8 (DML modification and retrieval statements) was not as strong as the others depicted in Figure 1 (i.e., the coupling measure between the associated subsets was lower). We choose to make it explicit, however, because it contributes to tie together the three subproblems concerned with different aspects of DML design. This may be the result of one of our personal biases as designers (i.e., we like to think of the DML as a logical unit) and, although it may have few implications for design, contributes to display the structure

of the design problem in a way more compatible with our perception of it, from a general standpoint. We think that there is nothing wrong with this, as long as (i) it corresponds to some extent to the results of the analysis (i.e., some coupling existed between the two subproblems), (ii) it is made clear that its implications are not as relevant as those of other relationships (note that we depicted this interaction as a dotted line in Figure 1) and (iii) why this is the case is well understood (in this case, we make it clear that the relationship merely indicates that statements in both groups should form a unified entity, the DML). If some of these circumstances fail to hold true in some situation (e.g., no coupling at all detected between the two subproblems), we would advise the designer to stop and think back trying to clearly understand the source of the mismatch between the results of the analysis and his personal intuition; for example, he or she may want, at this point, to reconsider his or her interdependencies' assessments between requirements in the two subsets. This constitutes a versatility characteristic of the methodology investigated here: while counterintuitive results may be obtained, thus breaking apart designer biases that may be unappropriate, the methodology also allows the designer to go back and study the reasons behind such results so as to completely understand why his biases are irrelevant to the problem at hand or, if such an understanding can't be achieved, it is still possible to modify the analysis accordingly.

So much for this slight digression. Going back to the central theme, the analysis of interactions between subproblems in the framework of

Figure 1, we come to subproblem 7, concerned with security and transaction recording, linked to subproblem 5. This interaction points out that some of the facilities used to support transaction recording can be used to deal with problems stemming from modification queries (e.g., cancelling a modification query may imply a back up that can use information from the record of processed queries); in addition, security should be more strictly enforced with modification requests.

Subproblem 5, playing the central role discussed in section 3.1 above, interacts with three more subproblems.Due to certain special characteristics of one of these interactions, we will postpone its discussion until the end of this section. The other two involve subproblems 3 and 1. While strong (the strongest, in fact, of the interactions in Figure 1, as measured by the associated coupling values), the interaction with subproblem 3, physical organization, is easily interpreted . It is clear that to ensure reasonable response time in the processing of modification statements, the adopted physical organization must be carefully designed. The new requirement 91 takes active part in the definition of this interaction (see Appendices B and F), pointing out that the modification of data items virtually maintained may result in considerable problems, more so if queries involving such items are of the critical type. In addition, note that although subproblem 8 wasn't nearly as related to subproblem 3 (we didn't even notice their interaction in Figure 1), the problems associated with efficient access methods are fully incorporated in the relationship between subproblems 5 and 3 (i.e., to modify a data item it must be located first).

The interaction between subproblems 1 and 5 is centered around the issue of maintaining logical relationships established by the schema (subproblem 1) upon the execution of modification queries (subproblem 5). This same issue was already identified in the first analysis; as a matter of fact, it has changed little from what it was before. Since the interfile relationships defined at the logical (schema) level are supported through correspondences among data items contained in the interrelated files, it is obvious that modifying the file contents can result in the destruction of the relationship, which would no longer be supported although the schema would still indicate its existence, the result being an inconsistent data base. Thus, this interaction emphasizes the fact that care should be taken to ensure that interfile relationships are maintained upon data base modification. This argument is complicated by the possible occurrence of "triggered" updates, since the same inconsistency outcome may result from the indirect modification of the contents of files involved; this is made apparent by the presence of requirement 22 (see Appendix A) in the materialization of this interaction (Appendix F). The appearence of the new requirement 90 (consistency constraints) in the links between these two subproblems re-enforces the same point: since updates can result in the violation of such constraints, defined at the logical level, they constitute another possible source of inconsistency that should be avoided when processing modification queries.

Subproblem 3, physical organization, interacts with several others, in addition to 5. Its relationship with subproblem 2, data items logical characteristics, emphasizes mainly issues of encoding for storage purposes For example, requirements 6, specifying ranges of values for data items, and 91, stating the possibility of virtual information, are involved in the definition of this interaction; they both suggest solutions to the encoding problem.

Its relationship with subproblem 10, concerned with compatibility requirements, makes explicit the fact that physical organizations strongly incompatible with those employed by other systems in the organization (the Government Agency in this case) would make the interaction with such systems more complicated.

Subproblem 3 also interacts with 11, reorganization. The requirements involved in this interaction point out the issue of flexible physical organizations, easily modifiable in response to performance degradation stemming from shifts in the query distribution.

The interaction between subproblems 10 and 2 is concerned about compatibility of logical characteristics of the data items across different systems.

Subproblem 11 interacts with subproblems 1 and 6, in addition to 3 as outlined above. The former is concerned with maintenance of the logical/physical mapping upon physical reorganization, that is, with data independence in the usual sense (note that the new requirement 92, which explicitly calls for this property, is involved in the relationship, see Appendices B and F).

The latter emphasizes that certain hardware characteristics called for by the set of requirements may facilitate reorganization (e.g., removable disks).

Subproblems 6 and 10 are also in interaction. This focuses on hardware characteristics that may be needed to make the eventual system compatible with others running presumably on other hardware.

Finally, we come to the interaction between subproblems 5 and 6 (DML modifidation statements and hardware characteristics). We postponed the discussion of this interaction until now because it is concerned with issues not as typically discussed in DBMS design as most of the ones involved in the interactions described above, thus being a bit more "original." It points out that some problems stemming from the need for supporting modification queries in a multi-user environment must be approached with certain hardware characteristics in mind. For example, since locks are needed to control concurrent modification queries, the specific lock assignment and usage facilities available can pose constraints on the way in which concurrent modification requests may be processed. Also, since the possibility of triggered modifications affecting files other than those explicitly stated in a query may result in complicating the deadlock problem (e.g., two concurrent queries that may seem completely independent at the outset may become involved in a deadlock due to the updates they trigger), the characteristics of recovery and back-up facilities become relevant to subproblem 5.

* * *

In summary, the interactions among the identified subproblems could all be reasonably explained and pointed out relevant design issues which were not apparent from looking at the overall requirements set as a whole. Some of them suggest possibly appropriate implementation strategies almost directly; others make some design trade-offs explicit which, while known to the designer (after all they were assessed in the early methodological steps), were not as clearly high-lighted at the outset.

The next section summarizes our experience with the application of the methodology investigated here and proposes a generalized methodology framework in which that experience is made explicit. It also introduces a number of areas for further research that at this point appear with great potential.

5. The Methodology in Perspective

In this section, an attempt is made to put our experiences with the application of the methodology in perspective. In particular, two main subjects are touched upon:

- the methodological steps as originally seen ([Andreu & Madnick 77]) are reconsidered in the light of our experience with them and coordination issues among steps to organize their execution are discussed; and
- several areas for further research are identified and briefly outlined.

5.1. Methodology Application Summary: A Framework

When the methodology investigated in this project was proposed (see [Andreu & Madnick 77]), its application was seen as a sequence of five steps leading to the identification of a design framework. Although the possibility of iterations was anticipated, the specific manner in which iterations might come about was not. The experience that we have gained through the actual application of the methodology to one concrete design problem has provided us with some information about when, how and why the need for iterations may emerge, so that we are in a better situation now to propose a generalized framework in which the different methodological steps are linked by means of a more elaborate scheme than a straightforward sequential execution of those five steps. Figure 2 schematically depicts such a framework.

The application of the methodology to a given design problem would start at the top of Figure 2. Circles in Figure 2 correspond to the 5 original methodological steps, other constructs are used to articulate coordination activities among these steps which we have identified in our experimental usage of the methodology.

Given an initial set of requirements, we proposed ([Andreu 77c]) a series of properties that its elements should possess (implementation independence, system structure independence, simplicity, etc.); in addition, we argued in section 2.2 above that requirements in that set

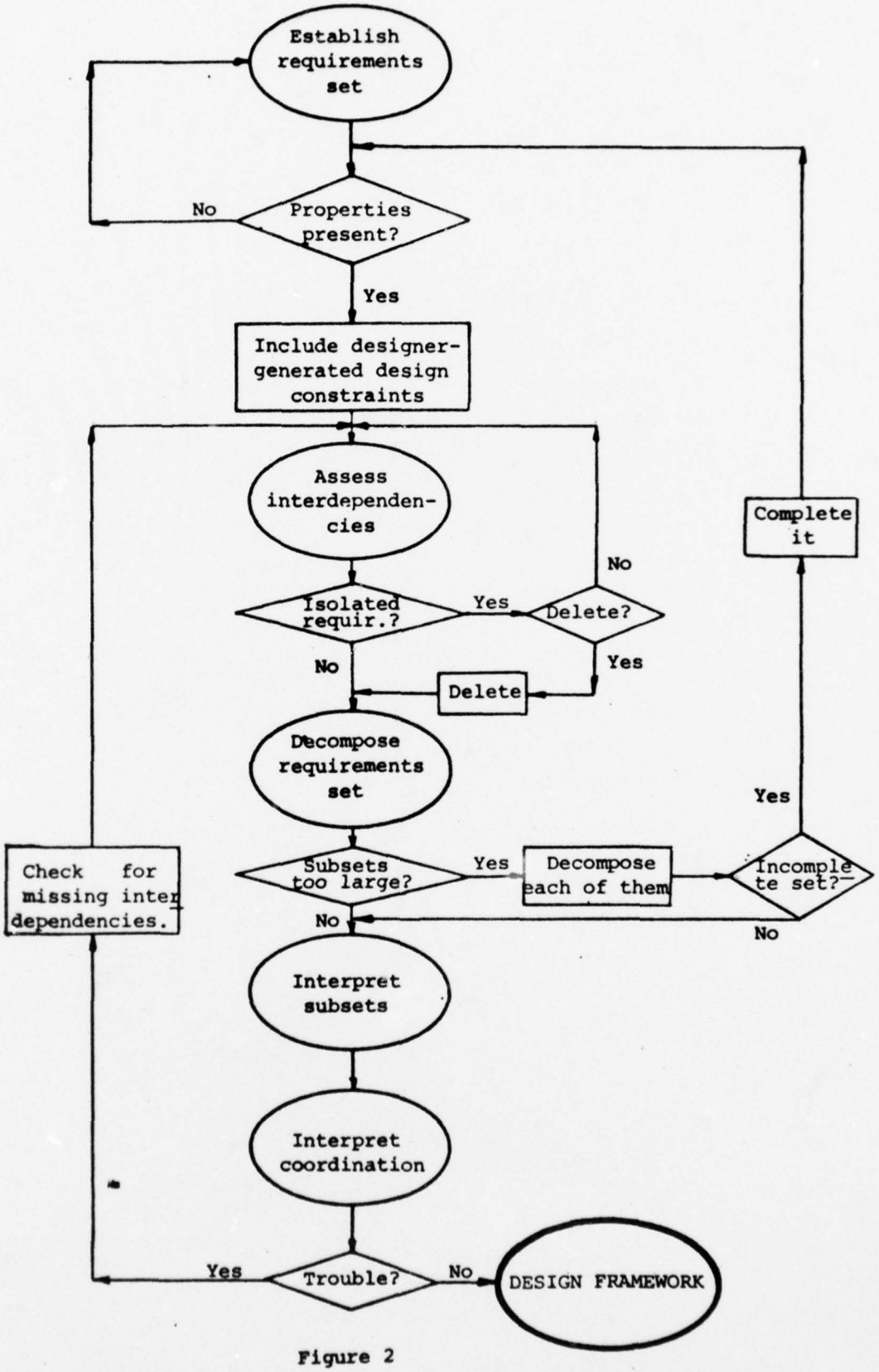

Establish requirements set
Properties present?
No
Yes
Include designer-generated design constraints
Assess interdependen-cies
Isolated requir.?
Yes
Delete?
No
Yes
Delete
Decompose requirements set
Subsets too large?
Yes
Decompose each of them
Incomple te set?
Yes
No
No
Check for missing inter dependencies.
Interpret subsets
Interpret coordination
Trouble?
Yes
No
DESIGN FRAMEWORK
Complete it

Figure 2

should be of comparable scope. Checking for the presence of such properties may imply the deletion, modification and/or decomposition of certain requirements, thus producing an updated set. The top part of Figure 2 makes this activity explicit.

Once the designer decides that the requirements set exhibits these properties, he may decide, before performing any analysis, that certain requirements are in fact missing, given his experience with similar design problems. This activity has not been explicitly performed in the two methodology applications that we have carried out, but it seems advisable given that the need for such requirements may appear later; its objective is thus to minimize the need for costly iterations. We see this activity as one in which the designer is likely to include what may be called "design constraints" in the requirements set; these constraints might reflect technological issues which the end-user(s) -- the main generator(s) of system requirements -- don't necessarily know about, but which nevertheless are important for design purposes.

At this point, the designer can proceed with the assessment of interdependencies among requirements. A series of guidelines to support this activity were proposed in [Andreu 77c].

Once interdependencies among requirements have been assessed and thus a graph-like structure imposed on the requirements set, looking for isolated nodes in the graph can suggest (i) that some interdependencies were overlooked or (ii) that certain requirements are in fact irrelevant to the problem and thus should be removed (recall that in [Andreu 77c] we

actually ran into such a situation). Either anomaly should be corrected before proceeding.

The resulting graph can then be decomposed using the techniques introduced in [Andreu 77a]. The resulting set partition should then be analyzed to determine whether some of them are too large so that a second decomposition step can be valuable as an indirect way to identify missing requirements (this was what happened in the analysis reported in [Andreu 77c]; recall, furthermore, that the relative importance of the coupling value associated with the partition measure can be, as suggested in section 3 above, an indication that a second decomposition should be performed). If the set of requirements is found to be incomplete, it must be completed in an activity that we described in section 2 of this report, and the steps discussed so far should be repeated.

If the decomposition obtained doesn't point out incompleteness problems, then each subset of requirements can be interpreted as a design subproblem and the links between subsets given subproblem coordination meaning. While doing so, the designer can run into trouble due to unexpected design problems and/or interactions. Going back to the original set of requirements and interdependencies should allow him to overcome the problems and clearly understand why the obtained results showed up, although he may be forced to change several of his pre-conceptions in the process. Once the different design subproblems and interactions are clearly understood, a design framework has been in fact identified. If, on the other hand, the designer can't understand or interpret the results, he may be able to identify interdependencies that were overlooked during the assessment process;

if this is the case, these interdependencies should be included and the new resulting graph decomposed, etc.

The two analyses that we have performed on the set of requirements originally introduced in [Andreu 77c] can be represented in the framework of Figure 2 as shown in Figures 3 and 4. Figure 3 represents the progress towards the design framework reported in [Andreu 77c], starting at the darkened circle; similarly, Figure 4 represents the activities performed to arrive at the design framework discussed in preceding sections.

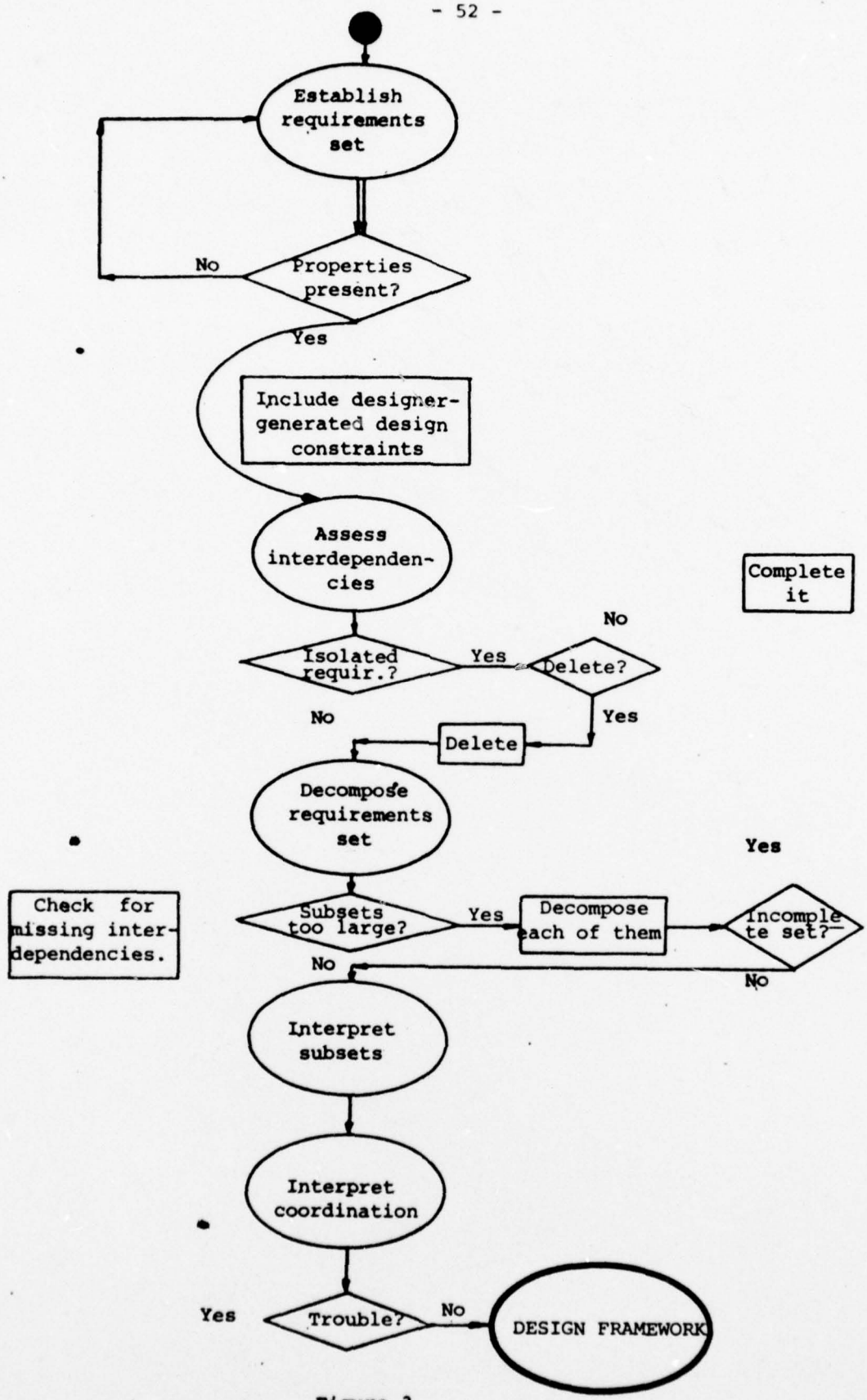

Establish requirements set
No
Properties present?
Yes
Include designer-generated design constraints
Assess interdependencies
Complete it
No
Isolated requir.?
Yes
Delete?
No
Yes
Delete
Decompose requirements set
Yes
Check for missing interdependencies.
Subsets too large?
Yes
Decompose each of them
Incomplete set?
No
No
Interpret subsets
Interpret coordination
Yes
Trouble?
No
DESIGN FRAMEWORK

Figure 3

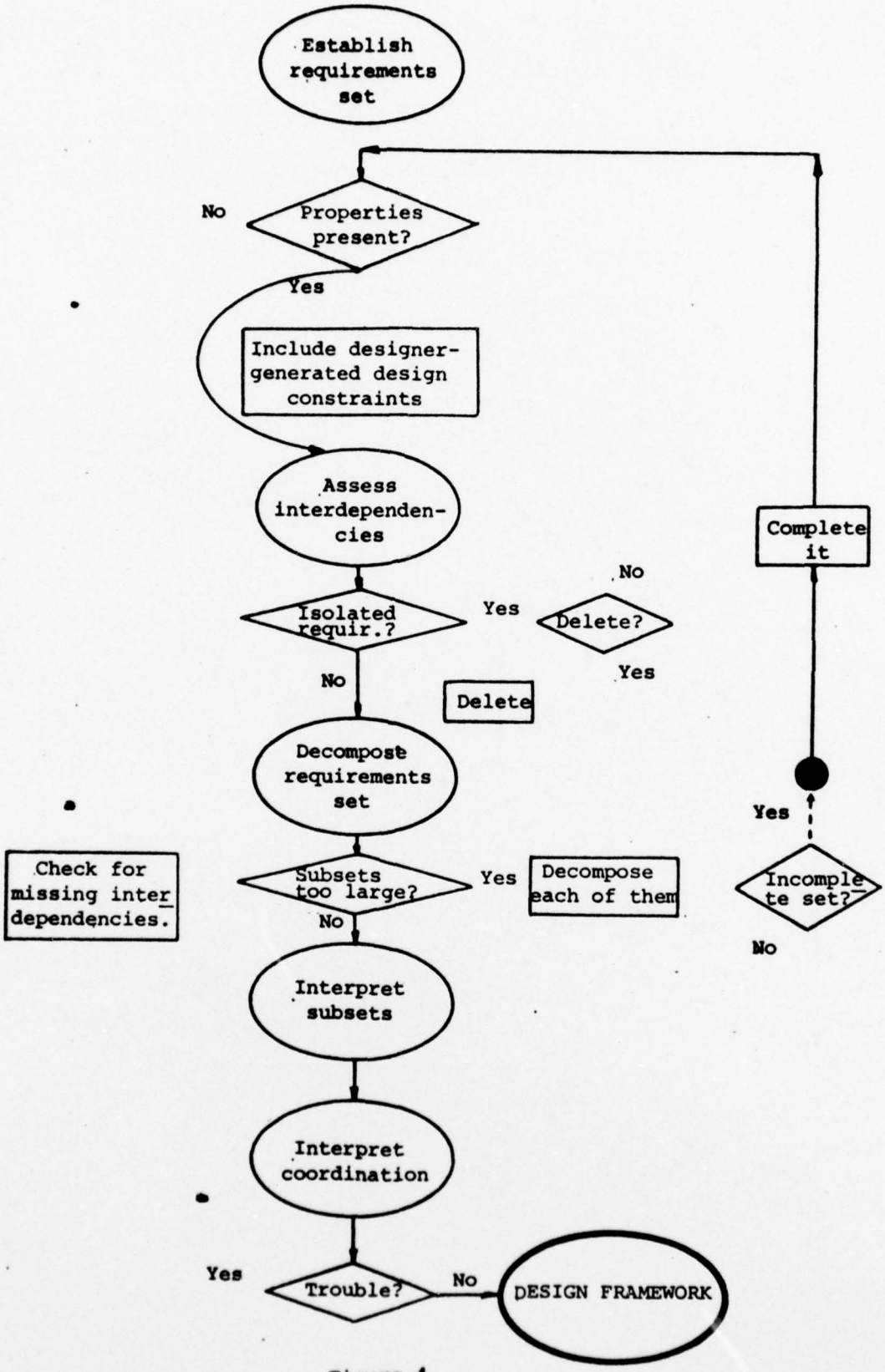

Establish requirements set
No
Properties present?
Yes
Include designer-generated design constraints
Assess interdependencies
Isolated requir.?
Yes
No
Delete?
No
Yes
Delete
Complete it
Decompose requirements set
Check for missing inter dependencies.
Subsets too large?
Yes
Decompose each of them
No
Yes
Incomplete set?
No
Interpret subsets
Interpret coordination
Yes
Trouble?
No
DESIGN FRAMEWORK

Figure 4

5.2. Areas for Further Research

Although the two methodology applications summarized above resulted in meaningful design frameworks for the problem we analyzed, there are several areas in which further research could add to the applicability of the methodology and to its effective inclusion into ordinary design practice.

The first of these areas is concerned with the explicit linkage of the activities propounded by the methodology to other design activities that must take place in the previous and following stages of the design process, the so-called "functional specification" and "detailed design" stages. The linking with the former can be possibly accomplished by means of using formal specification languages to establish the system requirements needed as primary input to the methodology investigated gated in this project. Doing so opens the possibility of relying on the logical constructs that such languages possess in order to try to automatize, to the extent possible, the interdependency assessment activity. At the other end, systematic procedures to move from design subproblems to specifications for software modules should be investigated in order to fully incorporate the methodology into the software design process.

Another area is that of refining the interdependency assessment process. The interdependencies used in this project have all been binary

(i.e., interdependencies were either present or they didn't exist at all). As a first approximation to investigate the kind of results that the methodology can generate this is acceptable and has worked reasonably well in the design problems we have tried to solve. However, from a general standpoint it is apparent that different "degrees" of interdependencies could be defined; even in a strictly subjective way one can come up with interdependencies of different importance. Trying to define a coherent interdependency scale and investigating procedures to assess interdependencies measured in that scale is a research activity which we believe can contribute significantly to obtaining more sensitive design frameworks. Several alternatives suggest themselves at the outset for this purpose. For example, characterizing each requirement by a set of design-related attributes would permit the computation of interdependencies as a function of the requirements' associated attribute values. Another possibility would be to use the logical and semantic constructs that a specification language may exhibit to deduce interdependencies in a systematic way; as briefly mentioned above, such a strategy can also contribute to a more explicit linkage of this activity to the activities in the preceding design process stage.

A third area which needs more work is that of supporting tools. We see the graph decomposition package described in [Andreu 77b] as a first step towards the construction of a comprehensive set of design support tools. Our experience already indicates, however, that it lacks a series of record keeping facilities which would enhance its usefulness significantly, particularly in subproblem identification and coordination.

Finally, we believe that more experiments should be conducted to

gain further insight into the usefulness of the methodology in real life problems. In particular, field studies in which different designers could be asked to use the methodology would be very useful to test it in real life design problems.

In summary, it is our contention that further research in these areas has great potential and should be undertaken.

REFERENCES

[Andreu 77a]: Andreu, R.C.: "Set decomposition: Cluster analysis and graph decomposition techniques", Technical Report #1, project N00039-77-C-0255, September 1977.

[Andreu 77b]: Andreu, R.C.: "Solving decomposition problems: Alternative techniques and description of supporting tools", Technical Report #2, project N00039-77-C-0255, September, 1977.

[Andreu 77c]: Andreu, R.C.: "An exercise in software architectural design: From requirements to design problem structure", Technical Report #3, project N00039-77-c-0255, December 1977.

[Andreu & Madnick 77]: Andreu, R.C. a-d Madnick, S.E.: "A systematic approach to the design of complex systems; application to DBMS design and evaluation", C.I.S.R. Report No. 32, M.I.T. Sloan School, March 1977.

[Huits 75] : Huits, M.H.H.: "Requirements for language in data base systems", in "Data Base Description", B.C.M. Douque and G.M. Nijssen, eds., North Holland 1975.

[Patterson 71]: Patterson, A.C.: "Requirements for a generalized data base management system", Procs. Fall Joint Computer Conference, AFIPS 1971.

APPENDIX A

INITIAL SET OF REQUIREMENTS

1) Data base schema (data dictionary) including interfile relationships, is defined and maintained independently of data base usage.

2) Separate files can be defined to be interrelated.

3) Data description language is English-like and self-documenting.

4) Data base schema is validated by system prior to usage.

5) Interfile relationships can be described at run time.

6) Field definition permits validation of input datum as to acceptable values.

7) The maximum number of files in the data base is at least 10.

8) Maximum number of interrelated files is at least 5.

9) Maximum size of a logical record is at least 500 information characters.

10) Maximum size of an item (field) is at least 100 characters.

11) Maximum number of items in a record is at least 50.

12) Repeated fields (multi-valued attributes) can be defined.

13) Variable sized fields can be defined.

14) Records in the data base can be added.

15) Records in the data base can be changed.

16) Records in the data base can be deleted.

17) Data base update can be performed by on-line user through query language.

18) Data base maintenance can be performed by batch.

19) A bulk data base update (initialization) can be performed through system utility.

20) Null-value field generation and identification supported.

21) Field update can trigger computation of a correlated field in same record.

22) Field update can trigger computation (e.g., tally of a correlated field in different record/file).

23) Data integrity supported at least at file level lockout on update.

24) Record level lockout.

25) Checkpoint/restore facilities.

26) Transaction history facility.

27) Separate security privileges for retrieval and update.

28) Data base level security.

29) File level security.

30) Field level security.

31) Non-procedural user's language available for on-line query and update.

32) Boolean conditionals can be used in record selection criteria.

33) Relational conditionals can be used in record selection criteria.

34) Arithmetic expressions can be used in record selection criteria.

35) Text string scanning expressions can be used in record selection criteria.

36) Relational condition can compare variable to variable.

37) Data meeting selection criteria can be used for subsequent query processing.

38) Selected data can be sorted by at least one sort key.

39) Capability for report formatting.

40) Report formatting optionally automatic.

41) Report break control feature supported.

42) Report summary line feature supported.

43) Multi-valued fields can be selectively listed.

44) Screen (menu) formatting facilities supported.

45) Local keyboard terminal supported.

46) Remote keyboard terminal supported.

47) CRT terminal supported.

48) Delta Data 5260 supported.

49) User can interrogate status of system.

50) User can interrogate status of current request.

51) User can cancel active request without loss of data integrity.

52) User can suppress listing, save report and later reinitiate listing.

53) User can direct output to a system printer.

54) User can route listing to other terminal.

55) Capability to broadcast messages to all terminals.

56) Signon Security.

57) A master terminal facility with privileged commands and control is supported.

58) Master terminal can be relocated to any on-line terminal.

59) System set-up effort and each subsequent SYSGEN less than one man-month.

60) Utilities to aid set-up.

61) Average up-time for the minimum configuration of at least 95% over a 30-day period.

62) Average system recovery time is 2 hours over a 30-day period.

63) Maximum recovery time is 24 hours.

64) Maintenance requirements less than 1 hour/week.

65) Power fail restart capability.

66) Dual processor fail soft capability.

67) Removable disks containing data base can be mounted and processed during an on-line session.

68) A job accounting recording facility is supported sufficient to charge users by application and by department.

69) A job accounting reporting facility.

70) Application is transportable to/from the Agency's existing systems.

71) Data is transportable to/from the Agency's existing systems.

72) System can communicate with other Agency's systems.

73) With a single terminal active, user can receive a response from a direct access to any item in the data base in less than 5 secs.

74) Response time as independent as possible of file size.

75) Capability to support at least 10 active terminals.

76) Capability to support two or more concurrent queries in different stages of processing.

77) Display rate of terminal at least 120 ch/sec on a CRT and 15 ch/sec on a hard copy terminal.

78) Dynamic file reorganization capability.

79) Dynamic reallocation of released and deleted storage areas.

80) File sizes are limited only by disk-storage capacities.

81) Total on-line data base can be distributed over many disk units.

82) Controls for tuning system performance at SYSGEN or system load time.

83) Controls for dynamically tuning system performance during run time.

84) Application tuning can be accomplished by restructuring the data to bias retrieval vs. update performance characteristics.

85) Can be delivered within M_1 months.

86) Non-recurring costs for basic configuration not more than C.

87) Maintenance costs less than MC/month.

APPENDIX B

List of New Requirements

88 Key fields (attributes) can be defined on records.

89 Logical operations can be performed on the logical structures (files) defined by the schema.

90 Consistency constraints involving records in different files can be defined.

91 Algorithmic relationships among data items can be defined that allow supporting virtual information.

92 Data dependency should be avoided.

93 Data redundancy should be avoided.

94 Different types of data items can be defined.

95 Depending upon its type, data items can be involved on certain types of operations (e.g., arithmetic, etc.)

96 Access to a record can be performed by specifying the value associated with its key attribute.

97 Certain queries are more critical than others and should tend to have better response time.

98 The DBA can control access authorization to the data base.

99 The DBA should have access to accounting information about all data base users.

100 Subschemas may be defined compatible with the data base schema.

101 Facilities for performance tuning should be under the control of the DBA.

102 If necessary for priority or efficiency reasons, the DBA can cancel and re-schedule requests in progress.

103 If necessary, the DBA can cancel printing and regenerate it later.

APPENDIX C

Interdependencies Involving New Requirements

(T Trade-off, C concurrency)

*Requirement 88 is related to:

·1 (T): Keys must be supported in the schema, and uniquely maintained.

·2 (C): Possibility of defining interfile relationships through keys.

·3 (T): Key definition supported by data description language.

·5 (C): Similar to (2) above.

·12 (T): Multi-valued attributes as keys may complicate processing.

·13 (T): Similar to above.

·89 (C): Facilitates definition of logical operations through keys.

*Requirement 89 is related to:

·1 (T): Schema definition must permit the specification of logical operations.

·2 (C): Possibility of compatibility, i.e., logical operations involving related files.

·4(T): Logical operations can be a source for validation.

·29(C): Logical operations involving entire files allow easy security checking at file level.

·31(T): Logical operations expressable in DML.

*Requirement 90 is related to:

·1(T): Constraints supported by schema.

·2(C): (2) can provide ways of checking the constraints.

·3(T): Constraints definable through DDL

·4(T): Incorporate constraint checking for schema validation purposes.

·14(T): Result should be checked against constraints

·15(T): Similar to (14).

·21(T): <u>All</u> results should be checked against constraints in this case.

·22(T): Similar to (21)

·23(C): Constraints permit integrity checking.

*Requirement 91 is related to:

·3(T): Relationships definable through DDL.

·11(C): (91) Provides alternatives to increase number of items in a record.

·21(C): Triggered computation may be suppressed if it affects a virtual data item.

·22(C): Similar to (21).

·23(C): Integrity checking simplified by (91).

·64(T): (91) may reduce maintenance time.

·73(T): Response time may increase if requested data items must be computed.

·74(T): Similar to (73).

·78(C): Possibility of maintaining items virtually increases number of alternatives for reorganization

·80(C): Way of reducing needed storage space.

·93(C): Maintaining items virtually avoids implicit redundance.

*Requirement 92 is related to:

·1(C): Compatible requirements.

·78(C): (92) makes it easier.

·81(C): (92) permits doing this more easily.

·82(C): Tuning can be performed independently of logical structure.

·83(C): Similar to (82).

·84(C): See (83).

·101(C): (92) permits the DBA to accomplish it with minimum disruption for the user.

*Requirement 93 is related to:

·2(C): Redundancy can be reduced through interrelated files.

·64(C): No redundancy permits simplification of maintenance.

·73(T): Access to data items may be more complicated.

·74(T): (93) can jeopardize it.

·81(C): Compatible requirements.

*Requirement 94 is related to:

·3(T): Different types definable through DDL.

·6(C): Different data types can be assigned different ranges in a generalized way.

·20(T): Null values' cahracterization may have to be different for different data types.

·95(C): Classification of operations facilitated by data types.

*Requirement 95 is related to:

·3(T): Definable (or maybe established by default) through DDL.

·20(T): Operations should be well defined on null values.

·31(T): Supported in DML.

*Requirement 96 is related to:

·31(T): Expressable in DML.

·73(C): Access through keys may facilitate this.

·74(C): Using access on keys may help to meet (74).

·97(C): Access through keys may be useful to solve critical accesses indirectly.

*Requirement 97 is related to:

·73(C): Compatible requirements.

·74(C): Similar to (73).

·76(T): Concurrency of critical queries a problem.

·78(C): May facilitate obtaining better response for critical queries.

·82(C): Provides a way to tune system according to critical queries.

·83(C): Similar to (82).

*Requirement 98 is related to:

·56(T): Under DBA control.

·57(C): Master terminal assigned to DBA can help to meet (98).

·58(T): Possibility to bypass DBA control.

·99(C): Authorization can be based on accounting information.

*Requirement 99 is related to:

·56(C): This defines a point in time which may be convenient for authorization control.

·68(T): General access restricted to DBA.

*Requirement 100 is related to:

·1(T): Subschema compatibility.

·2(T): Can pose constraints on possible subschemas.

·3(T): Subschemas definable through DDL.

·4(C): Associated facilities may be useful for subschema compatibility checking.

·27(C): Subschema definition time may be a convenient one to enforce security.

·28,29,30(C): Similar to (27).

·88(T): Pass keys along to subschema.

·89(T): Maintain logical operations on subschema.

·90(T): May limit possible subschemas.

*Requireemnt 101 is related to:

·82(T): Restricted DBA access.

·83,84(T): Similar to (82).

·92(C): Facilitates tuning.

*Requirement 102 is related to:

·41(C): Permits meeting (102).

·52(C): Similar to (41)

·55(C): User can be informed through this facility.

·103(C): Common processing possible.

*Requirement 103 is related to:

·52,53,54(C): Permits meeting (103).

·55(C): Permits informing the user.

APPENDIX D

STRUCTURE OF THE RESULTING GRAPH

NOLK

RECORDED LINKS.
FROM NODE TO NODE(S):

```
  1   ( 13)  2,  4,  5, 18, 19, 27, 59, 60, 88, 89, 90, 92,100,
  2   ( 23)  1,  5,  7,  8, 12, 14, 15, 16, 22, 23, 25, 29, 33, 51, 73, 76, 78, 82, 88, 89,
             90, 93,100,
  3   ( 14)  6, 12, 13, 20, 21, 22, 27, 31, 88, 90, 91, 94, 95,100,
  4   (  4)  1,  5, 89, 90,
  5   ( 11)  1,  2,  4,  8, 14, 15, 16, 17, 83, 84, 88,
  6   (  8)  3, 20, 21, 22, 23, 73, 74, 94,
  7   (  8)  2, 10, 19, 23, 27, 29, 76, 81,
  8   (  2)  2,  5,
  9   (  4) 10, 11, 12, 13,
 10   (  5)  7,  9, 11, 12, 13,
 11   (  5)  9, 10, 12, 13, 91,
 12   (  7)  2,  3,  9, 10, 11, 13, 88,
 13   ( 10)  3,  9, 10, 11, 12, 15, 20, 21, 22, 88,
 14   ( 17)  2,  5, 17, 18, 19, 20, 21, 22, 23, 25, 31, 51, 64, 74, 78, 79, 90,
 15   ( 10)  2,  5, 13, 19, 21, 22, 23, 25, 51, 90,
 16   ( 15)  2,  5, 17, 18, 19, 21, 22, 23, 25, 31, 51, 64, 74, 78, 79,
 17   ( 13)  5, 14, 16, 18, 19, 21, 22, 23, 25, 51, 64, 78, 79,
 18   (  7)  1, 14, 16, 17, 78, 79, 87,
 19   ( 13)  1,  7, 14, 15, 16, 17, 21, 22, 23, 25, 60, 62, 63,
 20   (  8)  3,  6, 13, 14, 31, 71, 94, 95,
 21   ( 15)  3,  6, 13, 14, 15, 16, 17, 19, 23, 25, 50, 51, 76, 90, 91,
 22   ( 16)  2,  3,  6, 13, 14, 15, 16, 17, 19, 23, 25, 50, 51, 76, 90, 91,
 23   ( 19)  2,  6,  7, 14, 15, 16, 17, 19, 21, 22, 25, 26, 27, 51, 65, 75, 76, 90, 91,
 24   (  3) 25, 75, 76,
 25   ( 14)  2, 14, 15, 16, 17, 19, 21, 22, 23, 24, 51, 62, 63, 76,
 26   (  3) 23, 51, 68,
 27   (  9)  1,  3,  7, 23, 28, 29, 30, 56,100,
 28   (  2) 27,100,
 29   (  5)  2,  7, 27, 89,100,
 30   (  2) 27,100,
 31   ( 25)  3, 14, 16, 20, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 43, 49, 50, 51, 52, 53,
             54, 73, 89, 95, 96,
```

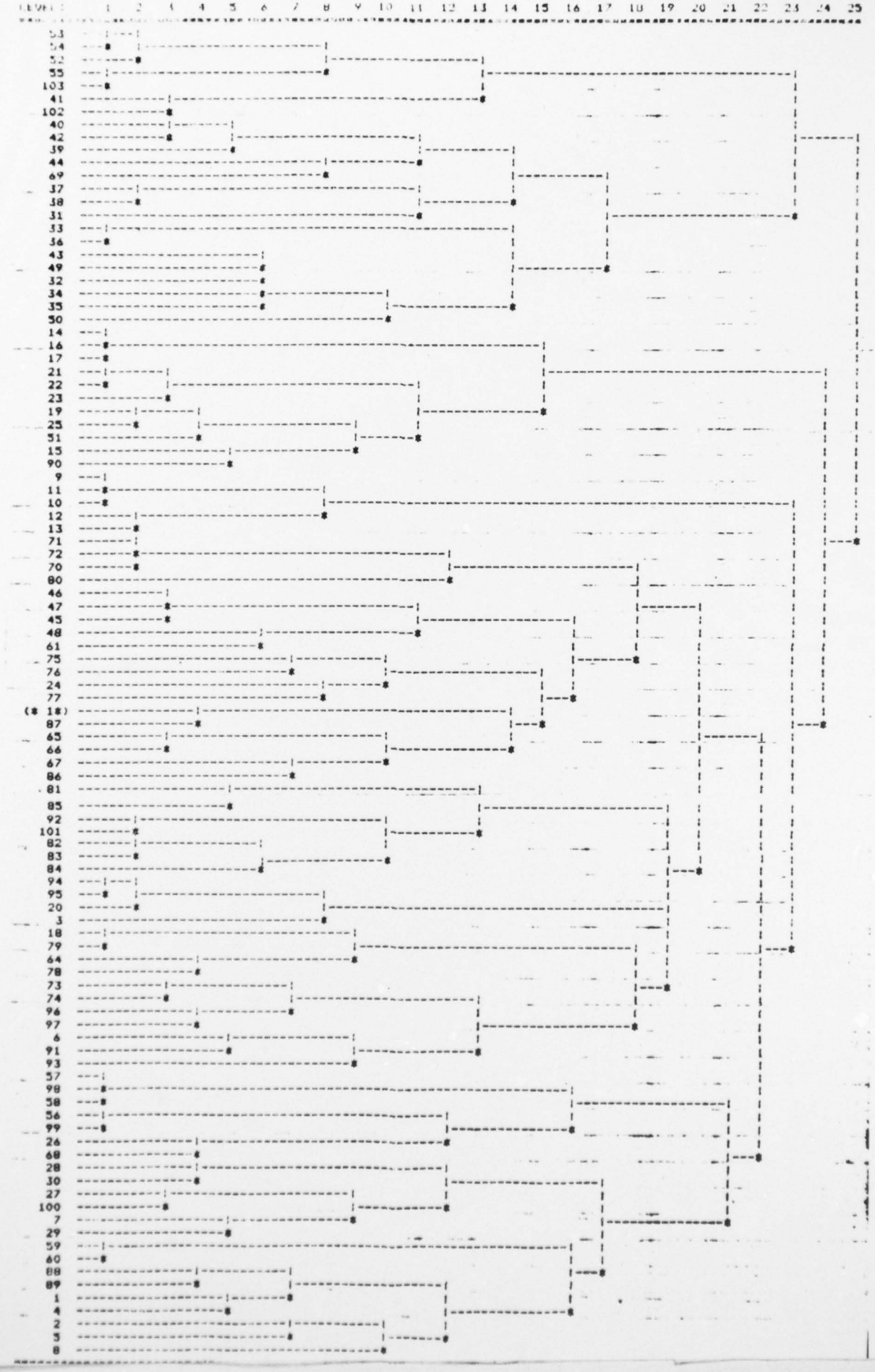
```
32   (  1) 31,
33   (  3)  2, 31, 36,
34   (  1) 31,
35   (  1) 31,
36   (  2) 31, 33,
37   (  3) 31, 38, 52,
38   (  3) 31, 37, 39,
39   (  6) 31, 38, 40, 42, 44, 69,
40   (  2) 31, 39,
41   (  3) 51, 52,102,
42   (  2) 31, 39,
43   (  1) 31,
44   (  1) 39,
45   (  2) 77, 86,
46   (  2) 77, 86,
47   (  2) 77, 86,
48   (  1) 86,
49   (  1) 31,
50   (  3) 21, 22, 31,
51   ( 13)  2, 14, 15, 16, 17, 21, 22, 23, 25, 26, 31, 41, 68,
52   (  7) 31, 37, 41, 53, 54,102,103,
53   (  4) 31, 52, 54,103,
54   (  5) 31, 52, 53, 55,103,
55   (  3) 54,102,103,
56   (  5) 27, 57, 68, 98, 99,
57   (  3) 56, 58, 98,
58   (  2) 57, 98,
59   (  2)  1, 60,
60   (  3)  1, 19, 59,
61   (  1) 86,
62   (  5) 19, 25, 63, 86, 87,
63   (  5) 19, 25, 62, 86, 87,
64   (  9) 14, 16, 17, 78, 82, 83, 84, 91, 93,
65   (  4) 23, 66, 67, 86,
66   (  3) 65, 76, 86,
67   ( 11) 65, 70, 71, 73, 74, 76, 81, 82, 83, 84, 86,
68   (  5) 26, 51, 56, 69, 99,
```

```
 69   (  2) 39, 68,
 70   (  5) 67, 71, 72, 80, 86,
 71   (  4) 20, 67, 70, 72,
 72   (  2) 70, 71,
 73   ( 11)  2,  6, 31, 67, 74, 75, 76, 91, 93, 96, 97,
 74   ( 11)  6, 14, 16, 67, 73, 78, 83, 91, 93, 96, 97,
 75   (  4) 23, 24, 73, 77,
 76   ( 12)  2,  7, 21, 22, 23, 24, 25, 66, 67, 73, 77, 97,
 77   (  5) 45, 46, 47, 75, 76,
 78   ( 12)  2, 14, 16, 17, 18, 64, 74, 79, 84, 91, 92, 97,
 79   (  6) 14, 16, 17, 18, 78, 80,
 80   (  3) 70, 79, 91,
 81   (  7)  7, 67, 82, 84, 85, 92, 93,
 82   ( 10)  2, 64, 67, 81, 83, 85, 86, 92, 97,101,
 83   (  8)  5, 64, 67, 74, 82, 92, 97,101,
 84   (  7)  5, 64, 67, 78, 81, 92,101,
 85   (  2) 81, 82,
 86   ( 12) 45, 46, 47, 48, 61, 62, 63, 65, 66, 67, 70, 82,
 87   (  3) 18, 62, 63,
 88   (  8)  1,  2,  3,  5, 12, 13, 89,100,
 89   (  7)  1,  2,  4, 29, 31, 88,100,
 90   ( 10)  1,  2,  3,  4, 14, 15, 21, 22, 23,100,
 91   ( 11)  3, 11, 21, 22, 23, 64, 73, 74, 78, 80, 93,
 92   (  7)  1, 78, 81, 82, 83, 84,101,
 93   (  6)  2, 64, 73, 74, 81, 91,
 94   (  4)  3,  6, 20, 95,
 95   (  4)  3, 20, 31, 94,
 96   (  4) 31, 73, 74, 97,
 97   (  7) 73, 74, 76, 78, 82, 83, 96,
 98   (  4) 56, 57, 58, 99,
 99   (  3) 56, 68, 98,
100   ( 10)  1,  2,  3, 27, 28, 29, 30, 88, 89, 90,
101   (  4) 82, 83, 84, 92,
102   (  4) 41, 52, 55,103,
103   (  5) 52, 53, 54, 55,102,

(AVERAGE NO. OF LINKS PER NODE:     6.447).
```

APPENDIX E

BEST GRAPH DECOMPOSITION


LEVEL 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
53
54
52
55
103
41
102
40
42
39
44
69
37
38
31
33
36
43
49
32
34
35
50
14
16
17
21
22
23
19
25
51
15
90
9
11
10
12
13
71
72
70
80
46
47
45
48
61
75
76
24
77
(* 18)
87
65
66
67
86
81
85
92
101
82
83
84
94
95
20
3
18
79
64
78
73
74
96
97
6
91
93
57
98
58
56
99
26
68
28
30
27
100
7
29
59
60
88
89
1
4
2
5
8

```
 REQ:
PRCL

 CLUSTER (NO) OBJECTS
 --------------------

    1    (15)   1   2   4   5   7   8  27  28  29  30  59  60  88  89 100
    2    ( 4)   3  20  94  95
    3    (11)   6  18  64  73  74  78  79  91  93  96  97
    4    ( 5)   9  10  11  12  13
    5    (11)  14  15  16  17  19  21  22  23  25  51  90
    6    (16)  24  45  46  47  48  61  62  63  65  66  67  75  76  77  86  87
    7    ( 7)  26  56  57  58  68  98  99
    8    (16)  31  32  33  34  35  36  37  38  39  40  42  43  44  49  50  69
    9    ( 7)  41  52  53  54  55 102 103
   10    ( 4)  70  71  72  80
   11    ( 7)  81  82  83  84  85  92 101

 REQ:
EVAL

 STRENGTH:   3.2673,
 COUPLING:   1.4886,
 MEASURE:    1.779.

 REQ:
```

APPENDIX F

INTERDEPENDENCIES AMONG THE SUBGRAPHS LISTED IN APPENDIX E.

FRLK

LINKS BETWEEN CLUSTERS 1 & 2 :
27 - 3
88 - 3
100 - 3

LINKS BETWEEN CLUSTERS 1 & 3 :
1 - 18
2 - 73
2 - 78
2 - 93

LINKS BETWEEN CLUSTERS 1 & 4 :
2 - 12
7 - 10
88 - 12
88 - 13

LINKS BETWEEN CLUSTERS 1 & 5 :
1 - 19
1 - 90
2 - 14
2 - 15
2 - 16
2 - 22
2 - 23
2 - 25
2 - 51
2 - 90
4 - 90
5 - 14
5 - 15
5 - 16
5 - 17
7 - 19
7 - 23
27 - 23
60 - 19
100 - 90

LINKS BETWEEN CLUSTERS 1 & 6 :
2 - 76
7 - 76

LINKS BETWEEN CLUSTERS 1 & 7 :
27 - 56

LINKS BETWEEN CLUSTERS 1 & 8 :
2 - 33
89 - 31

LINKS BETWEEN CLUSTERS 1 & 9 :
NONE.

LINKS BETWEEN CLUSTERS 1 & 10 :
NONE.

LINKS BETWEEN CLUSTERS 1 & 11 :
1 - 92
2 - 82
5 - 83
5 - 84
7 - 81

LINKS BETWEEN CLUSTERS 2 & 3 :
3 - 6
3 - 91
20 - 6
94 - 6

LINKS BETWEEN CLUSTERS 2 & 4 :
3 - 12
3 - 13
20 - 13

LINKS BETWEEN CLUSTERS 2 & 5 :
3 - 21
3 - 22
3 - 90
20 - 14

LINKS BETWEEN CLUSTERS 2 & 6 :
NONE.

LINKS BETWEEN CLUSTERS 2 & 7 :
NONE.

LINKS BETWEEN CLUSTERS 2 & 8 :
3 - 31
20 - 31
95 - 31

LINKS BETWEEN CLUSTERS 2 & 9 :
NONE.

LINKS BETWEEN CLUSTERS 2 & 10 :
20 - 71

LINKS BETWEEN CLUSTERS 2 & 11 :
NONE.

LINKS BETWEEN CLUSTERS 3 & 4 :
91 - 11

```
LINKS BETWEEN CLUSTERS    3 &    5 :
                  6 -   21
                  6 -   22
                  6 -   23
                 18 -   14
                 18 -   16
                 18 -   17
                 64 -   14
                 64 -   16
                 64 -   17
                 74 -   14
                 74 -   16
                 78 -   14
                 78 -   16
                 78 -   17
                 79 -   14
                 79 -   16
                 79 -   17
                 91 -   21
                 91 -   22
                 91 -   23

LINKS BETWEEN CLUSTERS    3 &    6 :
                 18 -   87
                 73 -   67
                 73 -   75
                 73 -   76
                 74 -   67
                 97 -   76

LINKS BETWEEN CLUSTERS    3 &    7 :
NONE.

LINKS BETWEEN CLUSTERS    3 &    8 :
                 73 -   31
                 96 -   31

LINKS BETWEEN CLUSTERS    3 &    9 :
NONE.

LINKS BETWEEN CLUSTERS    3 &   10 :
                 79 -   80
                 91 -   80
```

```
LINKS BETWEEN CLUSTERS    3 &   11 :
                64 -   82
                64 -   83
                64 -   84
                74 -   83
                78 -   84
                78 -   92
                93 -   81
                97 -   82
                97 -   83

LINKS BETWEEN CLUSTERS    4 &    5 :
                13 -   15
                13 -   21
                13 -   22

LINKS BETWEEN CLUSTERS    4 &    6 :
NONE.

LINKS BETWEEN CLUSTERS    4 &    7 :
NONE.

LINKS BETWEEN CLUSTERS    4 &    8 :
NONE.

LINKS BETWEEN CLUSTERS    4 &    9 :
NONE.

LINKS BETWEEN CLUSTERS    4 &   10 :
NONE.

LINKS BETWEEN CLUSTERS    4 &   11 :
NONE.

LINKS BETWEEN CLUSTERS    5 &    6 :
                19 -   62
                19 -   63
                21 -   76
                22 -   76
                23 -   65
                23 -   75
                23 -   76
                25 -   24
                25 -   62
                25 -   63
                25 -   76

LINKS BETWEEN CLUSTERS    5 &    7 :
                23 -   26
                51 -   26
                51 -   68
```

```
LINKS BETWEEN CLUSTERS    5 &    8 :
                  14 -   31
                  16 -   31
                  21 -   50
                  22 -   50
                  51 -   31

LINKS BETWEEN CLUSTERS    5 &    9 :
                  51 -   41

LINKS BETWEEN CLUSTERS    5 &   10 :
NONE.

LINKS BETWEEN CLUSTERS    5 &   11 :
NONE.

LINKS BETWEEN CLUSTERS    6 &    7 :
NONE.

LINKS BETWEEN CLUSTERS    6 &    8 :
NONE.

LINKS BETWEEN CLUSTERS    6 &    9 :
NONE.

LINKS BETWEEN CLUSTERS    6 &   10 :
                  67 -   70
                  67 -   71
                  86 -   70

LINKS BETWEEN CLUSTERS    6 &   11 :
                  67 -   81
                  67 -   82
                  67 -   83
                  67 -   84
                  86 -   82

LINKS BETWEEN CLUSTERS    7 &    8 :
                  68 -   69

LINKS BETWEEN CLUSTERS    7 &    9 :
NONE.

LINKS BETWEEN CLUSTERS    7 &   10 :
NONE.

LINKS BETWEEN CLUSTERS    7 &   11 :
NONE.
```

```
LINKS BETWEEN CLUSTERS    8 &    9 :
                  31 -  52
                  31 -  53
                  31 -  54
                  37 -  52

LINKS BETWEEN CLUSTERS    8 &   10 :
NONE.

LINKS BETWEEN CLUSTERS    8 &   11 :
NONE.

LINKS BETWEEN CLUSTERS    9 &   10 :
NONE.

LINKS BETWEEN CLUSTERS    9 &   11 :
NONE.

LINKS BETWEEN CLUSTERS   10 &   11 :
NONE.

REQ:
```